DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/91M-01

An Automated Red Player for the Theater Warfare Exercise

THESIS

Karl W. Kabanek
Captain, USAF

AFIT/GCS/ENG/91M-01

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1991	*Master's Thesis*

4. TITLE AND SUBTITLE
An Automated Red Player for the Theater Warfare Exercise

5. FUNDING NUMBERS

6. AUTHOR(S)
Karl W. Kabanek, Capt USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/91M-01

9. SPONSORING, MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Col Tom Yax
AU/CADRE/WG
Maxwell AFB, AL 36112-5532

10. SPONSORING, MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Theater Warfare Exercise (TWX) is a two-sided, theater-level, decision-making exercise created, maintained and used by the personnel at the Air Force Wargaming Center. It is used to allow military officers to practice the decision-making process needed for the wartime employment of air power.

An automated player was designed and a prototype implemented for the red (enemy) player using an expert system shell. The automated red player uses the TWX database that contains the data on the different units used in the exercise. From the data the automated red player builds mission packages for the various types of missions required in each day's activities by matching the day's requirements, given as prioritized target lists, and the characteristics of the aircraft available. The mission packages are not optimal but they are realistic and comply with the limitations placed on the red player. The output from the automated red player is in the same format as that currently used by the red players.

The automated red player reduces the amount of time needed to build the red mission packages by several hours. It also provides a standardized and realistic red player to allow comparisons of the success of different blue, friendly, teams.

14. SUBJECT TERMS
Expert System, Wargame, Automated Wargaming, Artificial Intelligence, TWX Application

15. NUMBER OF PAGES
159

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to **stay within the lines to meet optical scanning requirements.**

Block 1. Agency Use Only (Leave Blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency. Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ..., To be published in .... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availablity Statement. Denote public availability or limitation. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR)

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - DOD - Leave blank
DOE - DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports
NASA - NASA - Leave blank
NTIS - NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

AFIT/GCS/ENG/91M-01

An Automated Red Player for the Theater Warfare Exercise

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science (Computer Systems)

Karl W. Kabanek, B.S.
Captain, USAF

March, 1991

Approved for public release; distribution unlimited

Preface

The goal of this thesis was to design an automated red player for the ATAF portion of the Theater Warfare Exercise and implement a prototype of the automated red player. The design was developed which is flexible and efficient. The prototype was implemented with an expert system shell using the rapid prototyping process. This thesis lays the foundation for the complete automation of the red player for TWX.

I want to thank my advisor Maj Mark Roth for allowing me to do my own work and fight my own battles. I learned and accomplished much more with the independence he gave me than I would have if I had been led. I want to thank Dr Frank Brown for introducing me to the world of AI and the Brown Bravery Principal which allowed to boldly go where I had never gone before and usually didn't know how I got there or why I wanted to be there. Also, I appreciate the efforts of Maj Gregg Gunsch to help me organize my thoughts and put them on paper the way I meant them; his humorous comments on my drafts got the point across and made it interesting to read my thesis again and again.

The world would never have seen this thesis in its present form if it hadn't been for my three children, Anthony, Anna and Andrew. I couldn't have done it without them. They helped me and each other in countless ways. They understood when Dad was in a bad mood because the rules didn't work like I told them to. They made it a pleasure to come home after a hard day or night. We became more than family; we became friends. Thanks guys!

And finally thanks to Lt Col Messer for helping get to AFIT in the first place. He went out on a limb for me and made a big difference in my life.

Karl W. Kabanek

Table of Contents

List of Figures

List of Tables

Abstract

The Theater Warfare Exercise (TWX) is a two-sided, theater-level, decision-making exercise created, maintained and used by the personnel at the Air Force Wargaming Center. It is used to allow military officers to practice the decision-making process needed for the wartime employment of air power.

An automated player was designed and a prototype implemented for the red (enemy) player using an expert system shell. The automated red player uses the TWX database that contains the data on the different units used in the exercise. From the data the automated red player builds mission packages for the various types of missions required in each day's activities by matching the day's requirements, given as prioritized target lists, and the characteristics of the aircraft available. The mission packages are not optimal but they are realistic and comply with the limitations placed on the red player. The output from the automated red player is in the same format as that currently used by the red players.

The automated red player reduces the amount of time needed to build the red mission packages by several hours. It also provides a standardized and realistic red player to allow comparisons of the success of different blue, friendly, teams.

An Automated Red Player for the Theater Warfare Exercise

I. Introduction

1.1 Background

In 1976, the USAF Chief of Staff called for development of "...rigorous courses of study instructing operators and planners in the threat and application of force" (Air87:1) (original from USAF CoS CONSTANT READINESS TASKING, Item 6, 4 Aug 1976) to provide senior USAF officers the opportunity to practice the decision making skills needed in wartime. The senior officers needed a way to make wartime decisions and to evaluate the effect of those decisions in peacetime. This led to the creation of the Theater Warfare Exercise, TWX, in 1977.

TWX was designed in 1977 as "a four day, two sided, theater level, computer assisted, airpower employment decision making exercise" (Air87:1). The theater selected for TWX was NATO's central region in Europe since that was considered to be the most likely theater for our next major conflict. However, the lessons learned and the experience gained from the exercise would be applicable to any theater worldwide.

During the exercise, the participants make decisions characteristic of those an air component commander and his staff would be required to make during an actual war, including logistics and mission planning. The participants must decide how to allocate limited resources in order to accomplish the assigned missions. Real-world factors such as weather and facilities lost through enemy actions must be taken into consideration while making those decisions.

TWX has been an integral part of the Air War College curriculum since 1977. Each year it is played more than 80 times in numerous military courses such as the Contingency/Wartime Planning Course, the Combined Air Warfare Course and the Guard/Reserve Air Warfare Course. It is also used as part of the curriculum of the Royal Air Force Staff College and the Canadian Forces Command and Staff College (Har89:2).

1.2 Problem Statement

TWX keeps track of and manipulates thousands of variables and possibilities pertaining to aircraft characteristics, munitions characteristics, aircraft and troop movement, as well as the complex combat resolution functions. This has caused the management of the different aspects of the exercise to become very time consuming and difficult. During the exercise, an average red player, a member of the faculty representing enemy forces, will spend between five and eight hours per day implementing the red strategy and entering the required actions for the next day's play in the computer. This extensive effort prevents the red players from completely assimilating and analyzing all the information about the blue team's actions, decisions made by the student seminar representing the US or friendly forces, and providing the feedback the blue team needs to evaluate its decisions.

Personnel at the Air Force Wargaming Center participate in exercises for up to 20 different seminars or blue teams at a time. This necessitates the employment of 20 or more red players. Since each of the red players will respond differently to the scenarios presented, there is no standard by which the various blue teams can be compared.

The problem the Air Force Wargaming Center currently faces is the lack of manpower to adequately conduct the exercises. The Center needs a system which will meet the following goals:

1. Free Wargaming Center personnel from the computer interface tasks which currently consume the majority of the time spent conducting the exercise.
2. Develop a standardized, consistent and realistic red opponent for the blue teams to be evaluated against.

The first goal, that of freeing the red players from the computer interface tasks, needs to be accomplished since TWX is still a paper-driven exercise. For each day's actions a typical red player must request up to 32 reports, analyze their contents, complete 28 different worksheets and then transfer the data from the worksheets to the computer. Red players spend fifty to eighty percent of their time analyzing the data in the reports, filling out the worksheets and entering the data via a computer terminal (Har89:3) in the course of one ' 's turn. This massive amount of paperwork is the direct result of the organiza ı of the exercise. Each red player acts as the air commander of the two red fro.. ATAFs, the northern and southern fronts. The red player must make all decisions concerning all aspects of the red air war. Each category or decision area has its own worksheet and report.

The second goal, that of providing a standardized, realistic and consistent opponent for the blue teams, comes from the fact that each blue team plays against a different red player. This is necessary since each day's actions for a player consume so much time. Since each blue team plays against a different red player, each of whom has a different strategy, there is no reliable way to compare the results of the blue teams' actions.

1.3 Proposed solution

In order to meet these goals the Air Force Wargaming Center has requested that an automated red player be developed. An automated red player would reduce the amount of time the red players spend filling out the worksheets and free them to use that time to analyze the blue teams' actions and prepare an evaluation of those

actions. It would also allow the same red player to play against all the blue teams since it could be copied and used simultaneously by all the seminars.

An automated red player could take either of two forms: an object-oriented system or a system designed using artificial intelligence technology via an expert system shell. The latter was chosen based upon the nature of the problem. The major points which led to the selection of the expert system solution over an object-oriented approach are as follows, and are discussed in detail in section 3.2.

- The red player's decisions are based upon heuristics and rules of thumb rather than an algorithm.
- The solutions that may be found are not pre-enumerated. There are many different possible solutions to any given scenario.
- Experts are available who can solve the problem. This allows for consultation.
- The basic rules of reasoning for the system are stable.
- The system uses a large database which is easily accessible by an expert system shell.

A portion of this solution has been implemented by a previous thesis (Har89). The problems which remain were reduced to a single domain: automate the Allied Tactical Air Forces (ATAF) portion of TWX.

The ATAF portion of the exercise consists of two general actions, target selection and mission packaging. Each of these two areas can be further decomposed into the actions for each of the different mission types discussed in detail in Chapter IV.

1.4 Assumptions

This effort is based on the following assumptions:

- The previous work done toward building an automated red player is acceptable to the Air Force Wargaming Center.

- The Air Force Wargaming Center wants a system that allows targets to be entered which take precedence over those generated by the system itself.
- The solution should not be theater dependent, that is, it should be general enough to allow its use in different scenarios in different theaters worldwide.
- The system should allow the adding rules as needed.

1.5 Approach/Methodology

The approach used to developing the automated red player is both simple and straightforward: learn the manual system, design and implement the automated system. In the thesis proposal the following objectives were identified:

- Learn to play TWX using the manual method. Understand the reasoning behind the decisions made by red players by watching them play TWX and interviewing them.
- Design and implement the automated red player. The size of this effort depends upon the number and complexity of the rules needed to implement it. This should be done using the method known as rapid prototyping, explained in Section 3.4.
- Document the system. This includes the user's manual and the programmer's manual, included as appendices.

1.6 Materials and Equipment

The equipment used in this thesis was:

- One Sun 386i workstation.
- The expert system shell, Nexpert Object from Neuron Data.
- The Oracle Relational Database Management System.

This equipment was provided by the Air Force Wargaming Syster., Maxwell AFB, AL. Various manuals outlining the techniques and rules used in TWX were also provided.

1.7 Sequence of Presentation

Chapter II is a literature review of work and research currently being done in the area of automated wargaming. It provides an understanding of the current level of the technology related to this thesis and the background necessary to understand the research effort.

Chapter III explains the rationale behind the decision to use an expert system and the rapid prototyping methodology. Chapter IV describes the design of the automated red player. Chapter V presents a summary of the work accomplished and the recommendations for further research and work.

II. Literature Review

2.1 Introduction

Military wargaming is used worldwide to allow military commanders to practice strategies, maneuvers, tactics and decision-making skills under conditions which give them time to consider all options. This practice is also beneficial in that no actual materials are used and no lives are lost. Another advantage is the same scenario can be tried repeatedly using different approaches to answer "what if" questions. This allows military strategists to practice and evaluate their war skills at any time in preparation for the day when they are needed. Wargaming is the only practical method to allow the large number of officers in today's military to learn about strategy and tactics without actually going to war.

2.2 Automated Wargaming

Wargames are time consuming, calculation intensive exercises (Dav84:7). The utility of the wargame is quite frequently reduced because of the work involved in moving markers, calculating the results of battles, updating tables and performing many other bookkeeping functions. To combat this problem computers were introduced in 1954 when the Maximum Complexity Computer [illegible] was developed (All87:133). Since then computers have become an important wargaming tool. The use of computers in wargaming can be divided into three general categories: an automated combat resolution function, an automated player and an analytic wargaming system. Each of the three categories will be explained separately after a high level description of a generic wargame is presented.

2.2.1 A Generic Wargame Model. A wargame consists essentially of three components: the combat resolution system and two players, as shown in figure 1. The combat resolution system is the part of the wargame which determines the

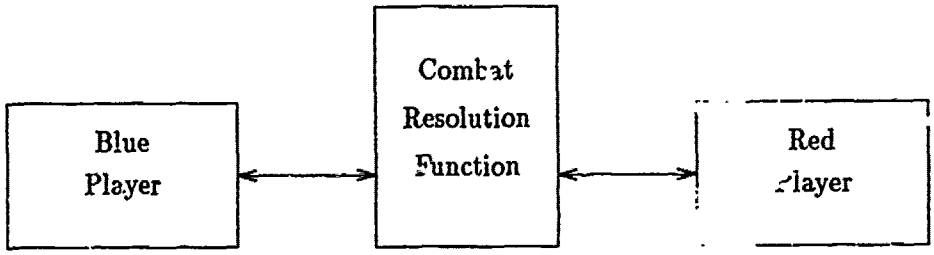

Figure 1. Generic Wargame Model

results of the actions of the players. If two opposing units attempt to occupy the same section of air or land in the scenario, for example, the combat resolution system determines the outcome of the ensuing dogfight or battle. The system may take into account such esoteric factors as morale and loyalty as well as the more concrete factors, such as relative unit sizes and armament.

2.2.2 Automating the Combat Resolution System. Traditionally, in the non-computerized, wargame the combat resolution system consisted of a set of tables used to compare the relative strengths and weaknesses of the units involved and dice to determine which row and/or column should be used in this particular encounter or battle. This was and is a slow and tedious process. In addition, inexact results quite frequently occurred because weapon capabilities and other factors were rounded because of the limited resources available and the need to simplify the manual calculation process.

In order to correct some of the shortcomings of the traditional combat resolution system brought on by the enormous number of variables involved, it was automated (Dav84:8). The automation of the combat resolution system involved converting the tables to a form usable by a computer program and caused the results to be much more exact and realistic. To further increase the correctness of the

combat resolution system, studies were conducted on the capabilities of weapons, weapon systems and troops under myriads of different situations. The results were then analyzed and reduced to equations and formulas which were then programmed and used in the appropriate wargames. This decreased the time needed to calculate the results of each player's actions and made the playing of the game easier.

In addition, the bookkeeping functions were handled automatically and the players no longer had to concern themselves with keeping track of the thousands of details necessary to make the wargame practical as a learning tool. This in turn allowed more detailed data to be used since the use of the computer made the corresponding increase in calculation time negligible. The end result was that the wargames were easier to use, faster and more realistic.

2.2.3 Automated Players. The motivation behind the automation of the players in a wargame is totally different from that used to justify the automation of the combat resolution system. One of the principal reasons for automating a player is the scarcity of qualified specialists available to play the role of the red (enemy) leadership. As Davis and Schwabe stated in (DS85:1)

> Red Agents can embody knowledge and concepts gathered from various sources over time. By contrast, it is usually difficult to put together a competent human Red Team because there are few specialists nationwide, and different specialists are needed to represent different levels of decision making.

By automating the red player the exercise no longer depends on the presence of a red player. The automated red player can be reproduced as many times as needed and used simultaneously for many different exercises (DBK86:2).

Another reason to automate the players in a war game is to better control the exercise. As people vary so do their responses to different situations which

arise during a war game. This makes the analysis of the responses to the various situations extremely difficult if not impossible. An automated player will react within a given range providing a more stable environment for the training being conducted (DBK86:2-3). This provides the human participants the opportunity to evaluate the effectiveness different strategies and decisions under controlled conditions.

2.2.4 Secondary Players. The automation of the players in a wargame provides the opportunity to add considerations and players which had not been considered previously. The introduction of secondary automated players acting as third world countries as well as other global powers has provided a new level of realism. The wargame is no longer a closed system involving only the two sides of the conflict; it can reasonably include automated players representing factions which might be dragged into the conflict and other parties which, while not directly involved in the conflict, might influence the outcome or conduct of the conflict (Dav84:9).

Adding secondary players allows the exercise to include considerations of how noncombatant parties might respond to the combatant's actions, e.g., the use of biological, chemical or nuclear weapons. This brings the exercise another step closer to accurately modelling the real world situation. It also serves to broaden the outlook of the human players. Instead of only considering what is happening between the combatants they must now consider what response their actions will cause among the noncombatants.

2.2.5 Analytic Wargames. When the combat resolution system and all the players involved in the exercise are automated in a wargame, it becomes an analytic wargame. It is called an analytic wargame because it can be used to analyze different scenarios involving the various players. Since the analytic wargame also simulates wartime conditions and actions it is sometimes referred to as a simulation (Dav88).

These analytic wargames are used to analyze many different possible scenarios which might arise during an actual conflict. Although a particular scenario might never occur during a war, the insights gained by observing the results of the simulations can provide military and political leaders the confidence to proceed with plans as they exist or to strengthen any weaknesses found. One of the major strengths of the analytic wargame is the ability to create various automated players for any side which reflect "its personality, grand strategy, and 'temperament' " (Dav88:17).

III. Evaluation and Selection of the Programming Methodology

3.1 Introduction

The automation of the ATAF portion of the Theater Warfare Exercise involved two distinct phases. The first phase was to determine what type of programming methodology should be used for the automation. This involved analyzing the requirements for the automated red player, comparing them with the various programming methodologies available and selecting the best methodology. The second phase was the actual design and development of the automated red player.

3.2 Selection of an Appropriate Programming Methodology

There are various programming methodologies available for any given problem. The choice of which methodology to use can and usually will have a great impact upon the development and the ease of maintenance of the implementation of the solution. The two methodologies considered for implementation of the automated red player were object-oriented programming using Ada and rule-based programming using an expert or knowledge-based system.

3.2.1 Object-oriented Programming Methodology. The object-oriented programming methodology is a well-known and accepted programming methodology. In this methodology the problem space is divided into objects and operations which act upon the objects and cause them to change their states. Each object has its own set of operations; an object influences other objects by sending messages to them.

The object-oriented methodology is best used when the main emphasis of the solution is the manipulation of numbers and data which can be represented as objects. This emphasis is best understood as being concerned with a specific set of objects and their states. The functions or operations used to change the states of the objects are contained within the description of the objects.

An object-oriented approach to the solution of the automated red player would be beneficial in the sense that it would eliminate the need for shared data areas. Also, each of the objects would be treated as an individual entity with its own operators which would virtually eliminate the possibility of incorrectly modifying the objects or tables.

However, the use of the object-oriented methodology has several very serious drawbacks. The first of these is that the decisions which will be made by the automated red player are not always straightforward "yes" or "no" decisions. These decisions would be difficult to model and implement using the methodology. Second, TWX is currently designed so that all the data about the numerous items involved in the exercise, such as air bases, aircraft, ground units and weather zones, are stored in a large database which is accessed and used by all the different programs supporting TWX. The database is an extremely efficient method of storing and accessing the data. Although the use of the object-oriented methodology does not preclude the use of the database, use of the methodology would unnecessarily complicate the work of designing and implementing the automated red player and necessitate reprogramming certain other functions already in use to ensure compatibility.

3.2.2 Rule-Based Programming Methodology. The rule-based methodology is a part of the artificial intelligence domain. In it the emphasis is placed on the rules which specify and bound the behavior of the system. The rules specify which actions are allowed and when they are allowed. After the set of conditions in a rule is evaluated, if all the conditions are satisfied, the corresponding goal is met and set to TRUE. If one or more of the conditions is not true, the goal is not met and the rule is set to FALSE. If the rule is set to TRUE, a set of actions associated with the rule are executed modifying data items and creating new ones as needed. This, in turn, can cause other rules to be put on the agenda or list of rules to be evaluated. This is a very flexible system which allows changes to occur based upon the evaluation of rules. Since many different conditions may exist which may cause a goal to be

evaluated to TRUE, there may be many different ways to meet a particular goal. This means that if a particular set of conditions does not meet a goal, the system will try a different set of conditions to see if the goal, as defined in the rule-base, can be met.

Another important consideration is that the rules and the data are separate. A rule-base can therefore operate on many different data-sets and come to different conclusions based upon their contents. When the database is changed, it does not affect the rule-base and vice versa. This independence also allows the rule-base to be modified as time goes on and new information about the system or the desired results becomes available.

3.2.3 Conclusion. Based on the emphasis of the two methodologies, summarized above, the rule-based programming methodology was tentatively selected as the methodology to be used. Once the tentative decision was made, a more in-depth analysis of the problem as it pertained to the rule-based methodology was accomplished.

3.3 Rule-based Methodology

There are several criteria which are used to determine if a particular problem is suitable for use with the rule-based methodology or an expert system. The criteria are divided into three categories: the type of problem to be solved, the availability of experts to provide the logic behind the solution and the management environment. Each category has several aspects which must be considered; the response for each category is usually given on a scale in addition to a simple yes or no answer (LDS89:3-17). The scale ranges from low, when the condition the criterion specifies is essentially or totally absent, to high, when the condition the criterion specifies is met.

3.3.1 Problem Criteria. The problem criteria area examines how well the particular problem to be solved lends itself to a solution using an expert system. Many problems, though they could be solved using an expert system, are better suited to other methodologies. The subcriteria and how each is rated for the automated red player are as follows:

- Does the problem generally involve symbolic reasoning? There is a large amount of math involved in the solution but it is more a bookkeeping function. The main emphasis is on the decisions to be made under the given conditions. Rating: moderate.
- Are test cases available? Yes, several test cases are available. Rating: moderately high.
- Is the problem well defined? Yes, the problem is very specific and is well defined. Rating: high.
- How frequently will the task be performed? It is performed at over 250 times a year. Rating: high.
- Does a written explanation of the solution exist? The conditions which bound the solution are contained in the Red Player Handbook (Air88). Rating: high.
- Does the task require only cognitive skills and not depend on common sense? Rating: high.
- Do the experts agree on the solutions to the problem? The experts at the Air Force Wargaming Center agree on the range in which the solution should be found. Rating: moderately high.

3.3.2 Expert Criteria. The expert criteria determine the quality of the assistance of the experts as it pertains to the given problem.

- Does an expert exist for the problem? Yes. Rating: high.
- Is the expert cooperative? Is the expert willing to assist in the development of the solution? The personnel at the Air Force Wargaming Center are available and supportive of the effort. Rating: high.
- Can the expert convey his knowledge competently? Yes. Rating: high.
- Is the expert's knowledge based on facts and experience? The experts have acted as red players at least ten (10) times. Rating: high.
- Does more than one expert exist? There are at least six red players which would be considered experts. Rating: high.

3.3.3 The Management Environment Criteria. The management environment criteria measure the support of the policy makers for the development of the expert system.

- Is there a need for the development of the expert system? Yes, as mentioned previously, the assignment of personnel as red players places a heavy burden upon the personnel of the Wargaming Center and keeps them from other beneficial duties. Rating: high.
- Is there sufficient financial support to see the development through to completion? The Air Force Wargaming Center has provided the equipment needed for the effort and continues to fund the necessary travel and expenses to continue the effort. Rating: moderately high.
- Does the top management support the development? Yes, They have provided the necessary funds and encouraged the experts to assist the effort. Rating: moderately high.

- Does management have realistic expectations about the use and usefulness of the final system? Yes, management does not expect the system to solve all their problems but to lighten the load of the red players. Rating: high.
- Will the users welcome the implementation and use of the system? Yes, there are the normal doubts and the "show me" attitudes but the users want the system to work. Rating: moderately high.

Another factor in favor of using an expert system is flexibility. During and after development of the automated red player the use of an expert system makes the modification of the rules and conditions much easier than would be the case in an object-oriented programming methodology where the algorithm would have to be rewritten. After evaluating the benefits of the flexibility offered, the answers to the questions about the different criteria and the ratings of the answers, I determined that the expert system solution was the better of the two alternatives and started the design and implementation of the automated red player (ATAF segment).

3.4 Rapid Prototyping.

In order to implement the automated red player as quickly as possible, the method called rapid prototyping was used. Rapid prototyping consists of repeating the evaluation, design, encoding and testing steps until the final solution is reached. In rapid prototyping, information is gathered from the expert and written sources and incorporated, as rules, in the expert system. The new system is then evaluated, tested and debugged. If the the system works according to the expert's explanation, it is accepted; if not it is refined until it matches the heuristics and knowledge the expert uses. At this point new knowledge is acquired and the process repeats itself until the expert system meets the specifications set for it (Ped89:182-185). This process is shown in Figure 2.

The use of the rapid prototype method allowed the automated red player to be built in small increments with each increment being tested and evaluated before

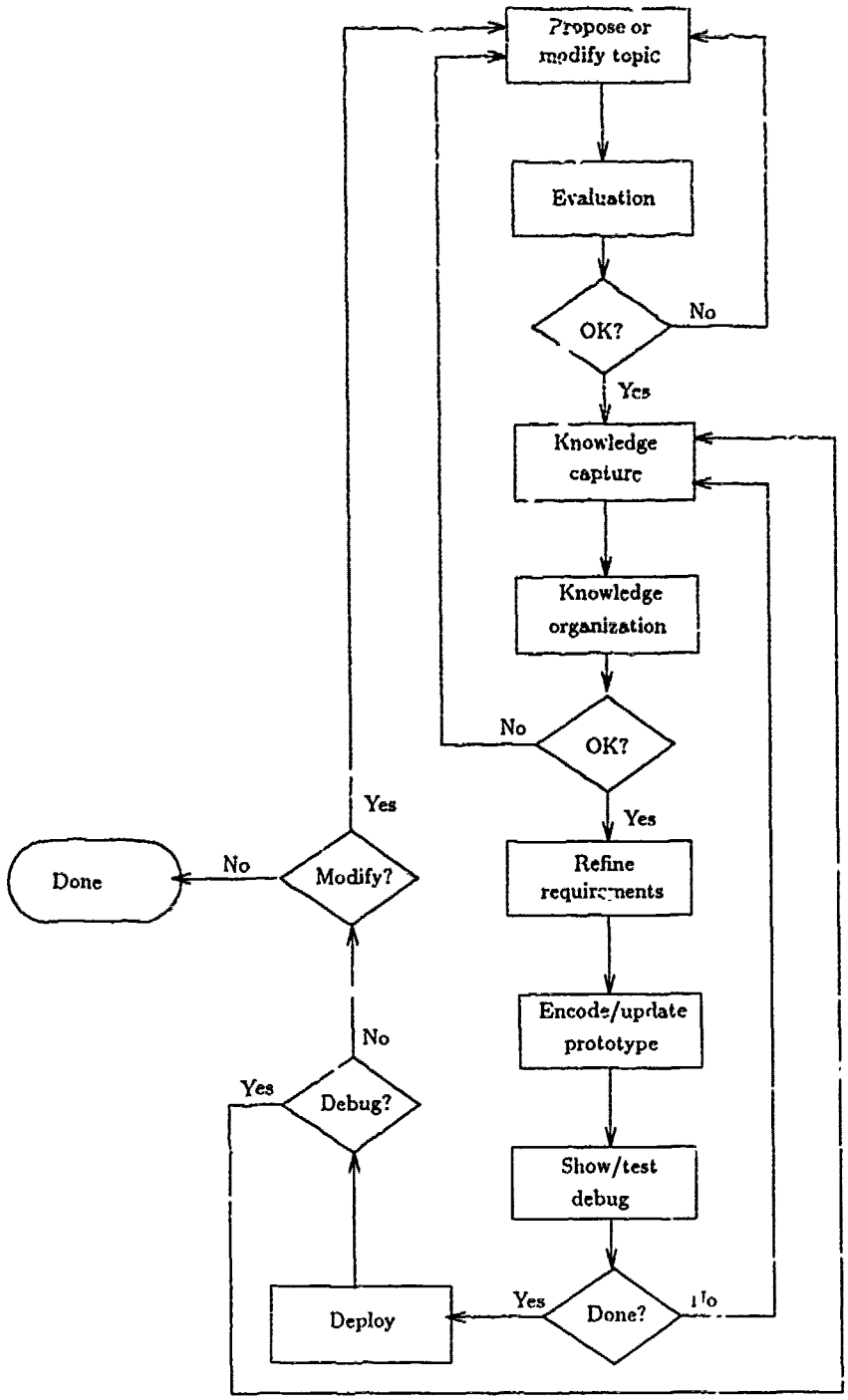

Figure 2. Rapid Prototype Process (Ped89:184)

additional knowledge is added to the system. It also allowed a partial rule-base to be used in the system. If a rule was refined or otherwise modified, the method ensured that extensive changes did not have to be made.

3.5 Conclusion

Although there are many methodologies which may be used to solve a given problem, one usually stands out as the best. In the case of the automated red player the choice was the rule-based methodology implemented using an expert system shell. The selection criteria all provide strong support for this decision. To further support the decision, the rapid prototyping process was examined and selected for use in the automation of the red player.

IV. Automation of the ATAF Portion

4.1 Introduction

The Allied Tactical Air Forces (ATAF) portion of the Theater Warfare Exercise (TWX) consists of several independent tasks. These tasks must be performed in a given order as specified in the Agile Eagle '88 Handbook (Air88:3.14-3.18) to ensure aircraft and other resources are used in the most advantageous manner possible. In addition the tasks must be accomplished for both day and night missions for both fronts. The tasks (in order) are given below and shown in Figure 3.

- Offensive Counter Air targeting, United Kingdom targets
- Area Defense Suppression allocation
- Offensive Counter Air targeting, other targets
- Interdiction and Battlefield Air Interdiction targeting
- Close Air Support allocation
- Electronic Counter Measures allocation
- Close Air Patrol allocation (not currently used)
- Defensive Counter-Air allocation
- Aviation Reconnaissance targeting

Currently TWX is organized around a conflict in the European theater. In order to make the automated red player applicable to any theater world-wide and allow it to be used even if the theater were changed, some parameters were made more general than they are currently. For example, the Offensive Counter Air (OCA) targeting was divided into long range OCA and regular OCA targeting instead of UK and other OCA targeting.

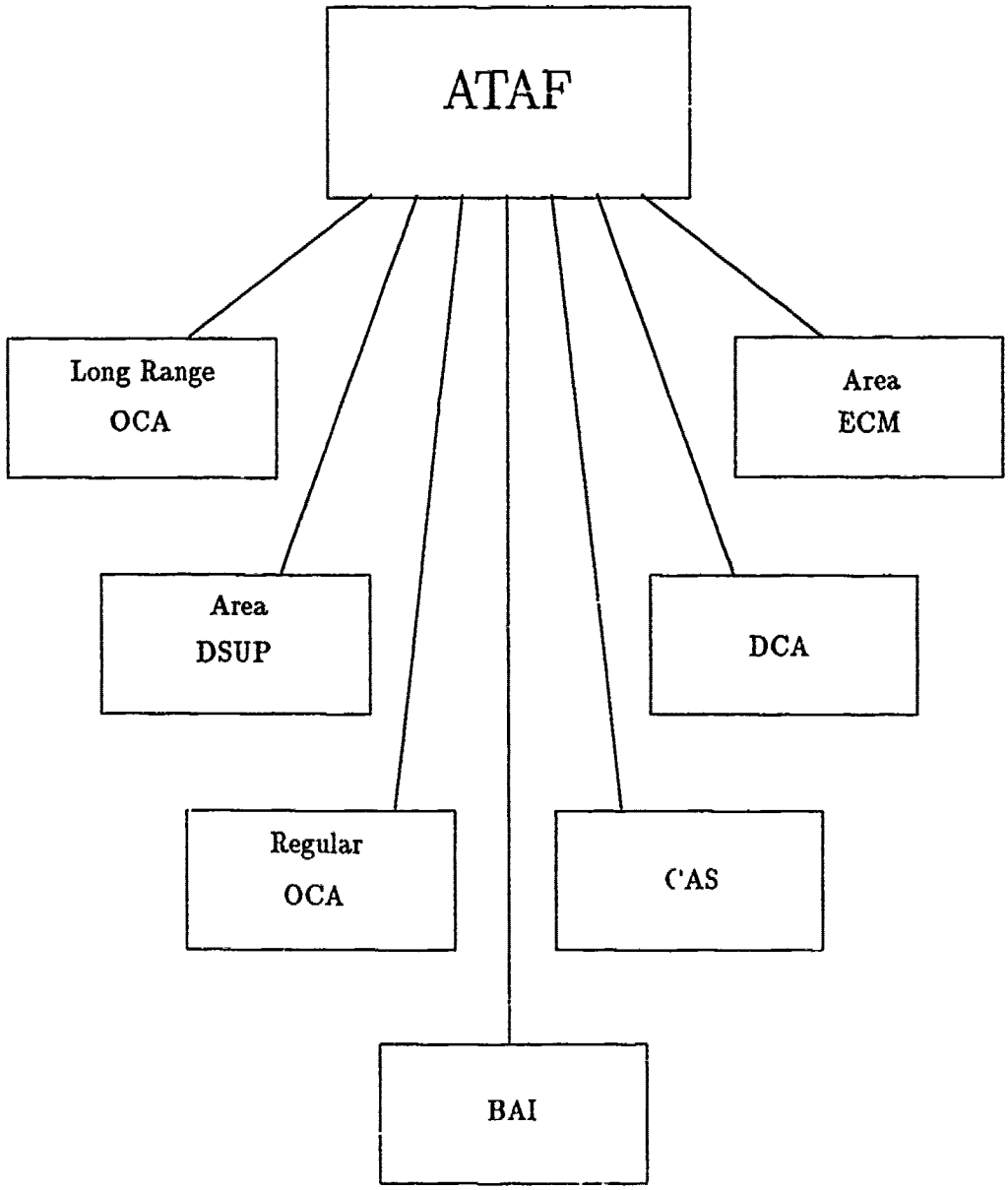

Figure 3. ATAF Top Level Design

4.2 Expert System Fundamentals

The automated red player was built using an expert system. To help the reader understand the design of the automated red player, a brief overview of the structure of the rules in the expert system is provided through an example.

In Nexpert Object a rule consists of three main parts; the conditions, the actions and the hypothesis or goal. Each rule has one or more conditions; if all the conditions of a rule are true the goal, hypothesis, of the rule is met. Whenever a goal of a rule is met, all the actions in the rule are executed sequentially. If the one or more of the conditions is false, the goal is not met and none of the actions are executed. However, several different rules may have the same goal or hypothesis. Therefore if a hypothesis is not satisfied by one rule the expert system will try to satisfy it by evaluating other rules. A goal or hypothesis is determined to be true if all the conditions of any one rule are true. A goal or hypothesis is determined to be false if at least one condition in each of the rules with that hypothesis is false. Therefore, a goal can be met if the conditions of one rule are all met, but it can be false only if none of the rules have all conditions met.

This example shows three rules used to determine what type of ECM aircraft should be used. All three rules have the same hypothesis, "Get_ECM_ac.Hypo" therefore the hypothesis or goal can be met by any of the rules having all its conditions evaluate to true.

When the first ECM aircraft are needed the type of ECM aircraft is unknown since none have been used. Therefore, the system must get the next type of ECM aircraft and the quantity of that type aircraft from the table. This rule retrieves this first type of ECM aircraft when the type of the ECM aircraft is unknown indicating the table has not been accessed.

```
CONDITIONS                          ACTIONS

Is  ECM.type   UNKNOWN        Retrieve  ECM.type & ECM.quantity
                                       from  lrecm table

HYPOTHESIS: Get_ECM_ac.Hypo
```

When there are not enough of the current type of ECM aircraft available and the principal aircraft have been selected, this rule retrieves the next type of ECM aircraft and outputs the table entry for the old ECM type.

```
CONDITIONS                          ACTIONS

ECM.quantity = 0                    ECM.index = ECM.index + 1
principal.quantity <> 0             Create table entry for old ECM type
                                    Retrieve ECM.type & ECM.quantity
                                        from  lrecm table

HYPOTHESIS: Get_ECM_ac.Hypo
```

If there are enough ECM aircraft of the current type available, no actions are needed since then system will get the aircraft from the type currently being used. The last rule has no actions (none are needed) and can still meet the goal. If there are enough ECM aircraft of the current type no actions are taken and the goal is satisfied by this rule.

```
CONDITIONS                          ACTIONS

ECM.quantity > 0

HYPOTHESIS: Get_ECM_ac.Hypo
```

Rules are the basic structure of the expert system but there is one other structure which must be explained. During the processing of the rule-base there are times

when a value of a specific object changes; this signifies that certain actions need to be accomplished. These actions are not part of a rule but are executed whenever the value of the object changes. Since the actions are not part of a rule they are located in the if-change metaslot of the object. This allows the object to act as a flag that is continuously monitored during the processing of the rule-base. Metaslots can be thought of as actions associated with a condition of "Has the value of this object changed?"

4.3 General Design Information

The various tasks involved in the ATAF portion of TWX were divided into several groups according to the similarity of function, decisions and design. The targeted mission group included both long range and regular OCA tasks, the CAS task and the BAI task, since they all require very similar decisions and targeting functions. The untargeted mission group included the area Defense Suppression (area DSUP), area ECM, and Defensive Counter Air (DCA) tasks because they do not have specific targets and the aircraft are assigned to fly in an undesignated airspace. Reconnaissance missions were designed as a separate category since they have different targeting requirements. The design for each of the three groups will be presented separately with any differences among the tasks being noted in each section.

Each of the different missions is affected by weather and daylight conditions. The discussion which follows explains the design of the general, good weather, daylight mission scenario. The main difference between this general case and the specific cases is essentially the destructive index of the aircraft allocated for the mission. This difference is accounted for by selecting the appropriate destructive index, according to the weather and daylight requirements of the mission, from the red aircraft table in the database and will be explained in detail later.

Before building the mission packages for the OCA, Close Air Support (CAS), Battlefield Air Interdiction (BAI) and Aviation Reconnaissance (RECCE) missions can begin, the targets for each mission type must be selected based upon the criteria developed for that type of mission. After the target selection is complete, the weather over each target must be determined as this may affect the type of aircraft selected for use against the targets (Air88:3.13). The specific target selection criteria for the various types of missions is discussed under the separate mission group headings.

4.4 Generation of Targeted Missions

4.4.1 Background. The targeted missions are, as the name implies, those types of missions generated against specific blue targets, such as airbases or ground units, or against a corps of a designated army for CAS. These mission types include long range and regular OCA, CAS and BAI. All these types of missions are packaged against a specific target and use at least one type of accompanying aircraft except CAS aircraft, which fly unaccompanied.

Long range offensive counter air missions are targeted against blue airbases and depots which have nuclear storage facilities or third generation aircraft and are beyond the range of regular OCA aircraft. These missions are designed to limit or eliminate the opponent's nuclear strike capability. This is accomplished by destroying the nuclear weapons in storage, incapacitating the delivery aircraft and damaging airbase runway systems. The principal strike aircraft are accompanied by long range ECM aircraft.

Regular OCA missions are targeted against the same type of target as the long range OCA missions with the same objective. The regular OCA packages include the principal strike aircraft, ECM aircraft, escort aircraft and may or may not include defense suppression aircraft depending on the strike aircraft type selected.

The BAI missions are targeted against blue ground units and which would hamper the advance of red ground units. These targets include "means of nuclear

attack aviation at the nearest airfields, tanks and artillery, strong points, centers of resistance, and river crossings" (Air87:2.19) as well as control centers and reserves. The mission packages are built of the same components as the regular OCA mission packages.

CAS missions are assigned to support red ground units and are targeted against enemy (blue) reserve units, artillery, mission installations and tanks. In TWX, the actual target chosen will be the Army corps in the area of interest. The red units receiving support from CAS missions are usually those "ground units which have penetrated enemy defenses and are conducting pursuit and exploitation operations deep within the enemy's zone of operations" (Air88:2.9).

To simplify the explanation of the building of targeted mission packages, the explanation will deal with one of the mission types; exceptions will be mentioned as necessary. The most general mission type is the regular OCA mission, therefore it was chosen for the explanation.

4.4.2 Solution. OCA missions are the first mission packages built and are assigned against targets throughout the theater. The long range OCA missions are assigned against targets out of range of standard aircraft, e.g., targets in the United Kingdom. The process of building an OCA package involves:

- building the target priority table;
- building the OCA aircraft tables;
- allocating the proper aircraft for the package; and
- creating the OCA mission table.

The organization of the OCA package building process is shown in Figure 4. Each of the subtasks depends upon and influences the subtasks listed before it. Although the subtasks are processed from left to right as given in Figure 4, a later task can cause an earlier task to be processed again under certain conditions.

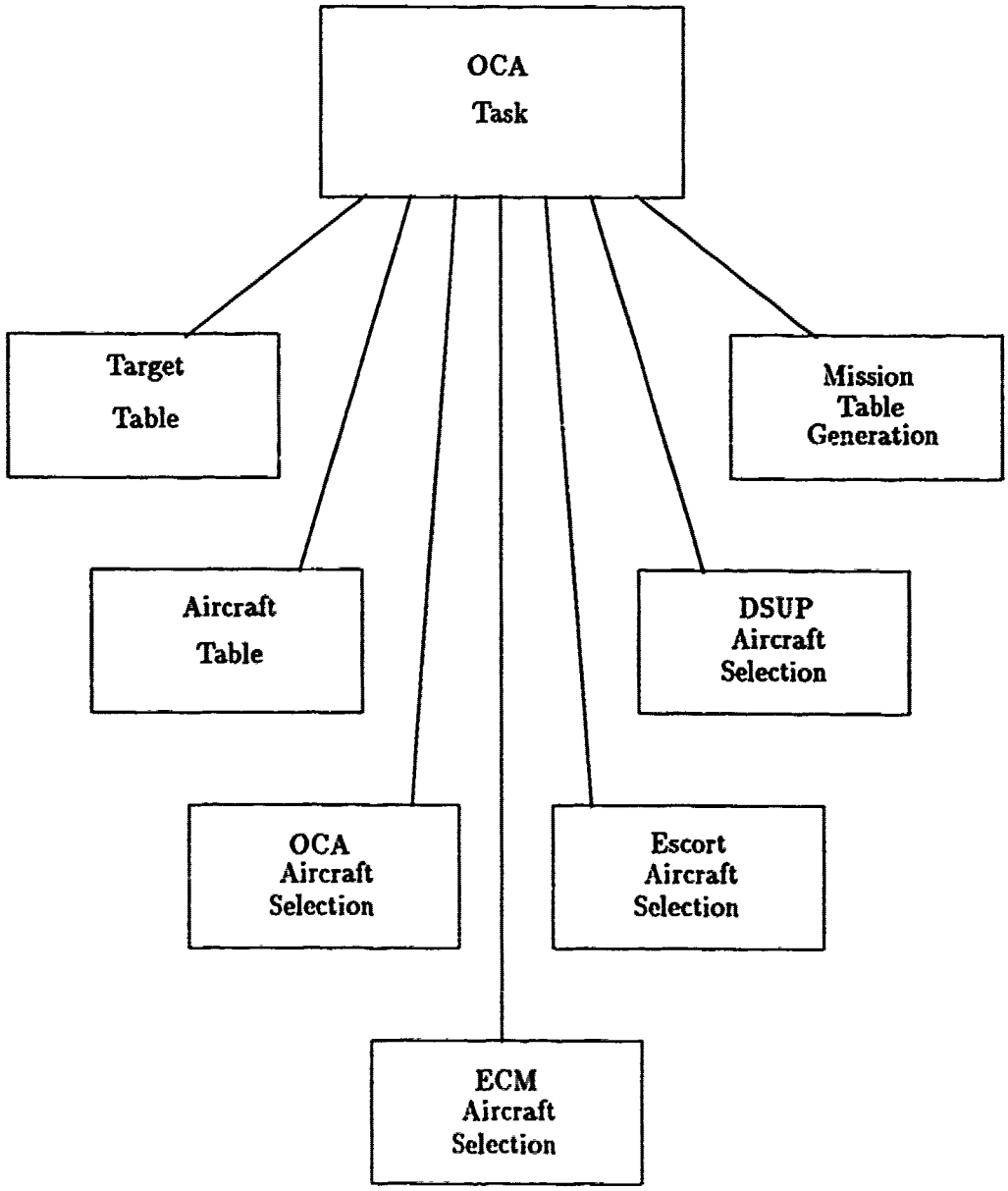

Figure 4. Organization of the OCA Process

4.4.2.1 Target Selection. Before the OCA missions can be planned by the automated red player all available OCA targets must be placed in a prioritized list according their potential for damage against the red player. I developed the following equation to calculate a target's potential for damaging the red forces, it's measure of merit:

$$\text{msrmrt} = \sum_{i=1}^{n}(\text{acqnty}\ (i) \cdot \text{acdi}\ (i) \cdot \text{wxcp}(i) \cdot \text{acsurvfac}(i))$$

where the following definitions hold:

- *msrmrt* is the measure of merit for the specified target;
- n is the number of different types of nuclear capable aircraft at the base;
- *acqnty* is the quantity of a particular type of aircraft;
- *acdi* is the destructive capability of the type of aircraft under consideration;
- *wxcp* is set to 1.1 if the type of aircraft under consideration is not all-weather capable, 1.2 if it is (based upon the increased utility of the aircraft); and
- *acsurvfac* is the survivability factor of the type of aircraft or what percentage of the aircraft can be expected to arrive on target if allowed to launch.

This equation takes in consideration the factors which bear upon a targets ability to damage red units in the exercise.

The building of the CAS target table involves finding the blue army corps in contact with advancing red units which have penetrated enemy defenses. The emphasis is placed on the army corps with units capable of stopping the red advance, e.g., artillery and tank units, missile installations and reserves. Since an army covers a wide area, targeting is done on the corps as a whole. The measure of merit of an army is determined by how many units are in contact with red units: this reflects the success of the red unit involved in that the further it has progressed, the more blue units will be called upon to engage and stop it

Table 1. BAI Target Table

nuclear delivery systems	5
command and control	3
air defense site	4
artillery	4
reserve units	2

The building of the BAI target table involves finding the rear echelon blue units which are advancing on red units. This type of mission emphasizes air defense sites and command and control points. Their measure of merit is based on the type of unit under consideration as shown in Table 1

After the measure of merit is calculated for each possible target, the targets are entered into a table in descending order of the measure of merit. This places the highest priority targets at the top of the list, ensuring that the most effective aircraft resources will be allocated against them first.

To illustrate the target prioritizing process, consider an example where there are three types of strike aircraft based at eight bases. The aircraft are identified as AC1, AC2 and AC3 with the relevant characteristics shown in Table 2. Each base has different quantities of each type stationed at it. These aircraft generate a given number of sorties each day. Applying the measure of merit equation and the data provided the measure of merit for each base can be calculated as shown in Table 3. These data are used to build the target table shown in Table 4.

In the prioritized target table the first column (index) is used by the expert system to distinguish and reference the individual records (rows). The second column (abid) is a unique identifier for the target used within TWX and the automated red player. Column three (status) gives the condition of the airbase rounded to the nearest 25% level to simulate the inexactness of reconnaissance data received during wartime. Column four is the calculated measure of merit for the target.

Table 2. Blue Aircraft Characteristic Table

AC Type	Destructive Index	Weather Capability	Survivability Factor
AC1	1.3	1.2	.95
AC2	.9	1.1	.85
AC3	.7	1.2	.80

Table 3. Blue Aircraft on Target Table

Target ID	AC Type	AC Quantity	AC msrmrt	Target msrmrt
40	AC1	100	148.20	148.20
41	AC1	42	67.43	
	AC2	36	30.29	
	AC3	104	69.89	167.61
42	AC1	82	121.52	
	AC2	20	19.20	
	AC3	20	16.83	157.55
43	AC2	130	109.40	109.40
80	AC2	98	65.86	65.86
82	AC1	60	88.92	88.92
84	AC2	80	53.76	
	AC3	80	67.32	121.08
85	AC1	79	117.08	117.08

Table 4. Prioritized Target Table

index	abid	status	msrmrt
1	41	1.00	167.61
2	42	.75	157.55
3	40	1.00	148.20
4	84	.50	121.08
5	85	.25	117.08
6	43	1.00	109.40
7	82	.75	88.92
8	80	1.00	65.86

4.4.2.2 Mission Aircraft Tables. The next step is to find all the aircraft which can be used to strike against the targets and place them in the aircraft tables. Since ECM, escort and defense suppression aircraft may accompany the primary OCA aircraft to help them survive the missions, four tables must be built, one to contain the data for the OCA aircraft and the others to contain the data for the accompanying aircraft. The aircraft are counted from all the eligible airbases and placed in the table with the aircraft with the highest good-weather, daytime destructive index (didg) at the top of the table. Examples of the four tables are given in Tables 5 and 6. In these tables the first column (index) is the field used by the expert system to access the various aircraft records. In the second column (actyp) is found the type of aircraft with the aircraft role at the end. The number of aircraft available is contained in the third column (acqnty). The rest of the columns (didg through dinp) contain the destructive index or mission effectiveness for the aircraft type for the particular type of mission (OCA, escort, DSUP or ECM) as stored in the TWX aircraft data tables. The third letter in the header refers to either day (d) or night (n) missions. The last letter refers to the weather conditions over the target, good (g), poor (p) or bad (b).

Table 5. OCA Aircraft Table

index	actyp	acqnty	didg	didp	didb	ding	dinp	dinb
1	T26A	57	1.10	.90	.70	.90	.70	.60
2	T22A	44	.85	.65	.45	.65	.45	.35
3	T16A	39	.90	.70	.60	.70	.50	.30

Table 6. ECM Aircraft Table

index	actyp	acqnty	didg	didp	didb	ding	dinp	dinb
1	T16E	22	.50	.40	.30	.50	.40	.30

4.4.2.3 Principal Aircraft Selection. The selection of OCA aircraft and the number used depends on two variables: the effectiveness of the type of aircraft under consideration and the status of the target. The first decision made by the automated red player is to determine how many OCA aircraft are needed for the mission being packaged. The second decision (whether the requisite number of OCA aircraft are available) involves accessing the OCA aircraft table. Depending on the which set of conditions are met in the automated red player's rule-base the OCA aircraft table may be opened, the aircraft acquired from the current record, the aircraft acquired from a new record or the aircraft may not be acquired causing the mission planning to halt for this task. The design is shown in Figure 5.

The selection of the aircraft type to be used is determined by the order of the aircraft types in the aircraft table with the most effective aircraft at the top of the table. In TWX and the automated red player, the format of the mission table requires the targets using the same type of aircraft to be placed together. Therefore, the first decision to be made by the automated red player is how many aircraft are required to reduce the target to 25% effectiveness or less or, if it is already at 25%

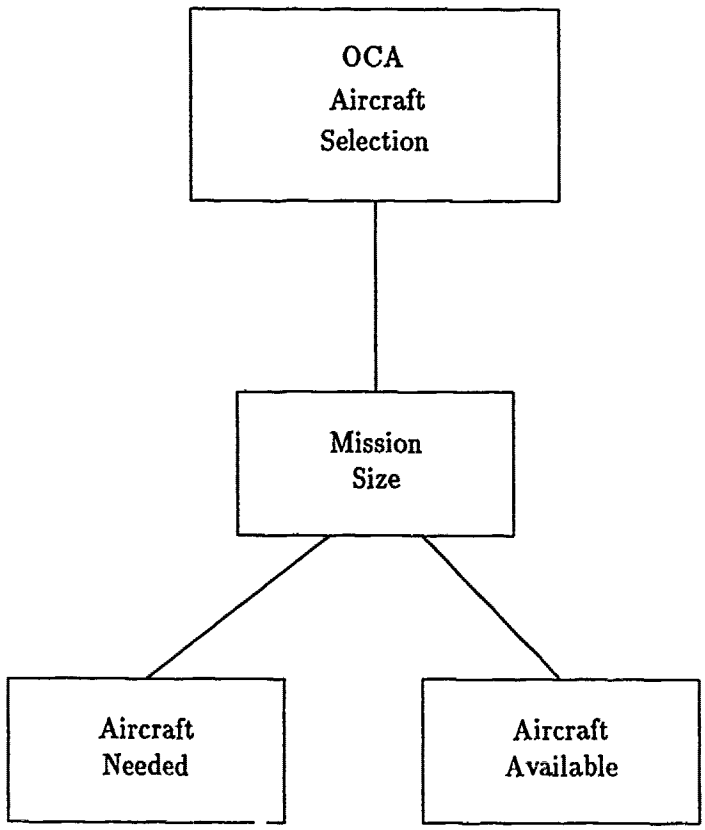

Figure 5. Design of OCA Aircraft Selection

Table 7. Damage Calculation Table

Number of Effective Sorties	Random Number 1	2	3	4	5	6	7	8	9	10
1 - 5	.5	.5	.5	.75	.75	.75	1.0	1.0	1.0	1.0
6 - 10	.25	.5	.5	.75	.75	.75	.75	.75	1.0	1.0
11 - 20	0	.25	.25	.5	.5	.75	.75	.75	.75	1.0
21 - 30	0	0	0	0	.25	.25	.5	.5	.75	.75
31 - 40	0	0	0	0	0	0	.25	.25	.5	.75
41 - 50	0	0	0	0	0	0	0	.25	.25	.5
51 - 60	0	0	0	0	0	0	0	0	.25	.5
61+	0	0	0	0	0	0	0	0	0	.25

effectiveness, how many aircraft are required to keep it at that level since the blue forces will be attempting to repair the airbase.

In order to determine how many aircraft are needed, I analyzed the damage calculation table from the TWX combat resolution model (Table 7). The damage calculation table contains the multipliers used to determine the status of a target after an attack. The status of a target after an attack is calculated by determining which row and column are used in Table 7 and multiplying the current status by the multiplier at the intersection of the row and column. The row is determined by calculating the number of effective sorties (number of aircraft in Table 7) over the target by multiplying the number of aircraft in the package by their appropriate destructive index. A random number is generated and used to determine which column is applicable.

I calculated the average multiplier for each level of quantity of aircraft. This was then used to calculate the expected damage to a target when the median number of effective sorties or breakpoint reached the target for each level of target status (25%, 50%, 75% and 100%). I generated an expected damage table for a 100% survival rate of the aircraft (Table 8), a 75% survival rate of the aircraft (Table 9)

Table 8. Expected Target Status, 100% Survival Rate

Number of Aircraft	Surviving Aircraft	Status Multiplier	Current Target Status 25%	50%	75%	100%
3	3	.78	.19	.38	.58	.78
8	8	.68	.17	.34	.51	.68
15	15	.55	.14	.28	.41	.55
25	25	.30	.08	.15	.23	.30
35	35	.18	.04	.09	.13	.18
45	45	.10	.03	.05	.08	.10
55	55	.08	.02	.04	.06	.08
66	66	.03	.01	.01	.02	.03

and a 50% aircraft survival rate (Table 10). After analyzing these three tables I set the desired number of effective sorties for each level of airbase status as shown in Table 11. These rates produce the desired damage to the targets, allowing for some aircraft losses without wasting valuable resources.

Finally, to determine how many aircraft of a certain type were needed against a specific target, the number of effective sorties needed for the target given its status was determined by using Table 11. That number was divided by the destructive index of the aircraft type as taken from the aircraft tables in the database. For example, using T22As with a destructive index of .85 against target 42 with a status level of 75%, the number of T22As needed is calculated by dividing the destructive index of the T22As (.85) by 25 (the desired number of effective sorties from Table 11) giving 29 aircraft needed. The expected status of the target after the mission would be between 41%, if 50% of the aircraft are lost, and 23%, if none of the aircraft are lost. In addition, up to 8 effective sorties could be lost without any degradation of the mission effectiveness since 21 effective sorties is the cut-off for this level (see Table 7).

Table 9. Expected Target Status, 75% Survival Rate

Number of Aircraft	Surviving Aircraft	Status Multiplier	Current Target Status 25%	50%	75%	100%
3	2	.78	.19	.38	.58	.78
8	6	.68	.17	.34	.51	.68
15	11	.55	.14	.28	.41	.55
25	19	.55	.14	.28	.41	.55
35	26	.30	.08	.15	.23	.30
45	34	.18	.04	.09	.13	.18
55	41	.10	.03	.05	.08	.10
66	50	.08	.03	.05	.06	.10

Table 10. Expected Target Status, 50% Survival Rate

Number of Aircraft	Surviving Aircraft	Status Multiplier	Current Target Status 25%	50%	75%	100%
3	2	.78	.19	.38	.58	.78
8	4	.78	.19	.38	.58	.78
15	8	.68	.17	.34	.51	.68
25	13	.55	.14	.28	.41	.55
35	18	.55	.14	.28	.41	.55
45	23	.30	.08	.15	.23	.30
55	28	.30	.08	.15	.23	.30
66	33	.18	.04	.09	.13	.18

Table 11. Required Effective Sorties

Target Status	Effective Sorties
100%	35
75%	25
50%	15
25%	5

Even though the status of the target would in actuality be a percentage such as 23% or 41%, and it would be stored in the database as a more accurate decimal number, the solution presented above is only concerned with the four levels of target status (25%, 50%, 75% and 100%) since reconnaissance cannot be totally accurate. This reflects the wartime situation and actual reconnaissance capabilities.

4.4.2.4 ECM Aircraft Selection. The selection of the ECM aircraft to be used for the package is not target-dependent. That is, the ECM function is basically the same regardless of the target. The determining factors are, simply, how many aircraft are needed and whether they are available. This is depicted in Figure 6.

The first decision made by the automated red player is to determine how many ECM aircraft are needed for the mission being packaged. Due to the nature of the ECM function and the red aircraft being used, that number has been set at one (Air88:3.14). This decision (one ECM aircraft per target) was implemented with rules to ensure that any future modifications could be made easily.

The second decision (whether the requisite number of ECM aircraft are available) involves accessing the ECM aircraft table. Depending on the which set of conditions is met in the automated red player's rule-base, the table may be opened, the aircraft acquired from the current record, the aircraft acquired from a new record, or the aircraft may not be acquired, causing the mission planning to halt for this task.

4.4.2.5 Escort and Defense Suppression Aircraft Selection. The selection of the escort and defense suppression aircraft to accompany the strike aircraft in a mission package follows essentially the same logic as the selection of the ECM shown in Figure 6. First, the number of aircraft needed for the mission is determined in relation to the type and number of strike aircraft selected. In general, there are 50% as many escort aircraft and 33% as many defense suppression aircraft as there

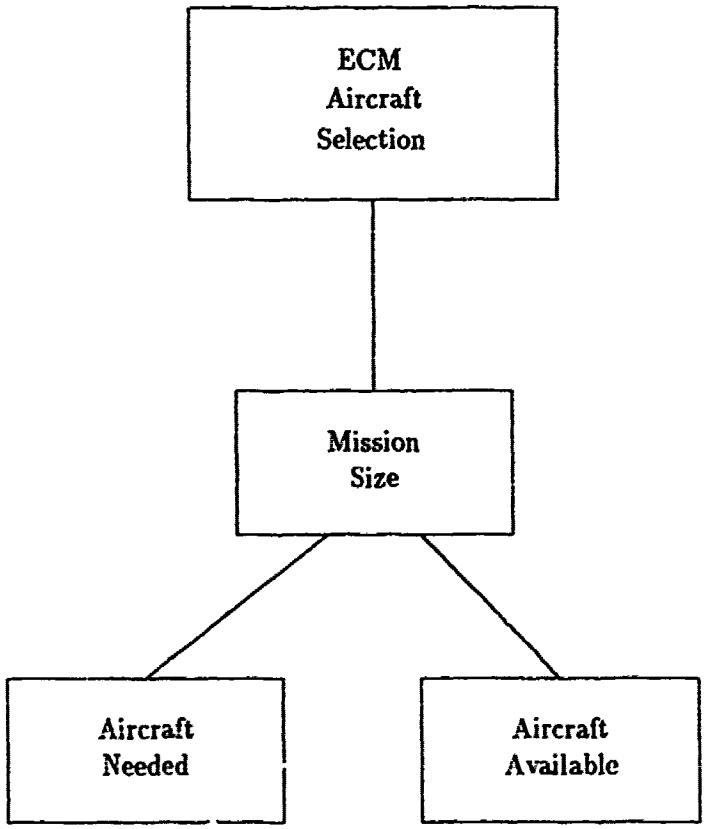

Figure 6. Design of ECM Aircraft Selection

Table 12. Possible Accompanying Aircraft, OCA Missions

Mission Conditions	Strike Aircraft	Escort Aircraft	DSUP Aircraft	ECM Aircraft
long range/day	T16A/T22A T26A	none	none	T16E
long range/night	T26A	none	none	T16E
regular/day	U24A/M27A M23A	M23D M29D	M23A/M27A U24A/U17A	T16E Y28E
regular/day	T16A/T22A T26A	M25D M31D	none	T16E
regular/night	U24A/M27A M23A	M23D M29D	M23A/M27A U24A	T16E
regular/night	T26A	M25D M31D	none	T16E

are strike aircraft in each mission package. However, when certain types of aircraft are selected as the strike aircraft for a package, no defense suppression aircraft are required. Second, the type of accompanying aircraft is determined. Certain types of accompanying aircraft can only escort certain types of strike aircraft due to aircraft characteristics and capabilities. This relationship is summarized in Tables 12 and 13. Because of the nature of the CAS missions, no escort, ECM or defense suppression aircraft accompany the strike aircraft.

Table 13. Possible Accompanying Aircraft, BAI Missions

Mission Conditions	Strike Aircraft	Escort Aircraft	DSUP Aircraft	ECM Aircraft
day	U24A/M27A M23A/U17A	M23D	M23A/M27A U24A/U17A	T16E/Y28E
day	T16A/T22A T26A	M25D/M31D	M23A/M27A U24A/U17A	T16E
night	U24A	M23D	U24A	T16E
night	T26A	M25D/M31D	none	T16E

Table 14. Long Range OCA Mission Table

AC Type	AC Qnty	Tgt 1	Srt 1	Tgt 2	Srt 2	Tgt 3	Srt 3	Tgt 4	Srt 4	Tgt 5	Srt 5	ECM Type	ECM Qnty
T26A	12	41	12	0	0	0	0	0	0	0	0	T16E	1
T16A	108	42	28	40	39	84	17	85	6	43	18	T16E	5
T22A	47	82	30	80	17	0	0	0	0	0	0	T16E	1

If, during the allocation of aircraft for a mission package, there are too few ECM, DSUP or escort aircraft to complete the package either the number of aircraft in the package must be reduced or a lower priority target, which does not require as many aircraft, selected.

4.4.2.6 Mission Table Generation. The final task left to the OCA mission generation is the creation of the mission table, Table 14. The table format is the same as the form currently used by the red players. It provides the aircraft type for a line of missions with the quantity of aircraft needed for them and the type and number of ECM, escort and defense suppression aircraft needed. The rest of the table contains the target identifiers paired with the number of aircraft designated for that particular target. Up to five target-quantity pairs are included on a single line or record.

4.4.2.7 Summary. The OCA, BAI and CAS mission generation tasks are the bulk of the tasks accomplished under the ATAF portion of the automated red player. They require the generation of target and aircraft tables that are used to determine how many of each type of aircraft are to be assigned to each target. The results of these decisions are then output in a table in the same format currently used by the red players as shown in Table 14.

4.5 Generation of Non-Targeted Missions

4.5.1 Background. The non-targeted missions are missions assigned over a designated area without a specific target. There are two types of non-targeted missions, area Defense Suppression (area DSUP) and Defensive Counter-Air (DCA). Each of these types of missions involves only one type of aircraft, the strike aircraft. No other aircraft accompany the strike aircraft. These differences distinguish these missions from the targeted missions.

The area DSUP missions are built to suppress the enemy's air defense capability. That is, area DSUP aircraft attack the enemy or blue player aircraft. This allows friendly, red, aircraft to operate more freely and with fewer losses than would otherwise occur.

The DCA missions are designed to inhibit enemy air activity against red targets. The ultimate goal of the DCA missions is to "prevent the enemy from conducting reconnaissance and delivering attacks by aviation and pilotless aircraft on troops, naval forces, and targets of the rear" (Air88:2.9). This is accomplished by shooting down enemy aircraft during attacks.

4.5.2 Solution. Both the area DSUP and DCA missions are designed the same way. The only differences are the type of aircraft used and the number of aircraft used. For both types of missions an aircraft table is built in the same manner the aircraft tables are built for the targeted missions. Then aircraft are selected by types in multiples of ten (10). The area DSUP mission uses a total of 100 aircraft (Air88). The DCA mission, since it is the last mission package built using strike aircraft, uses all the defensive aircraft left (Air88).

After the aircraft are assigned, the mission table is created. The table simply contains the type of aircraft assigned and the number of that type aircraft. An example of the DCA mission table is shown at Table 15.

Table 15. DCA Mission Table

AC Type	Quantity	AC Type	Quantity	AC Type	Quantity
M21D	120	M23D	700	M29D	380

4.6 Generation of Reconnaissance Missions

4.6.1 Background. Reconnaissance missions are conducted against operational, strategic and tactical targets including airbases and ground units. The reconnaissance missions are extremely important because they not only locate potential targets, but also provided information on the condition and status of the targets. This information is used by the rule base to make decisions concerning how many of each type of aircraft were needed to inflict the required damage on a target.

In real-life there is no such thing as perfect intelligence. To model this in TWX, the true status of a target is not made available to the players: a range of values was represented by the status values. For example, a status value of 50% actually means the status of the target under consideration is greater than 37.5% and less than 67.5%. This allows for the difficulties in obtaining intelligence data and correctly analyzing it.

4.6.2 Solution. Before reconnaissance mission planning can begin, the targets have to be selected. The targets for reconnaissance missions were divided into three categories: known targets, suspected targets and unknown targets. Known targets are those conclusively identified previously, e.g., airbases. Unknown targets are those whose predicted location was reasonably accurate but whose capabilities were not accurate, e.g., mobile ground units not actually moving. Suspected targets are targets which may or may not be at the predicted location, e.g., advancing mobile ground units. Each of the targets is put in an appropriate table according to its category.

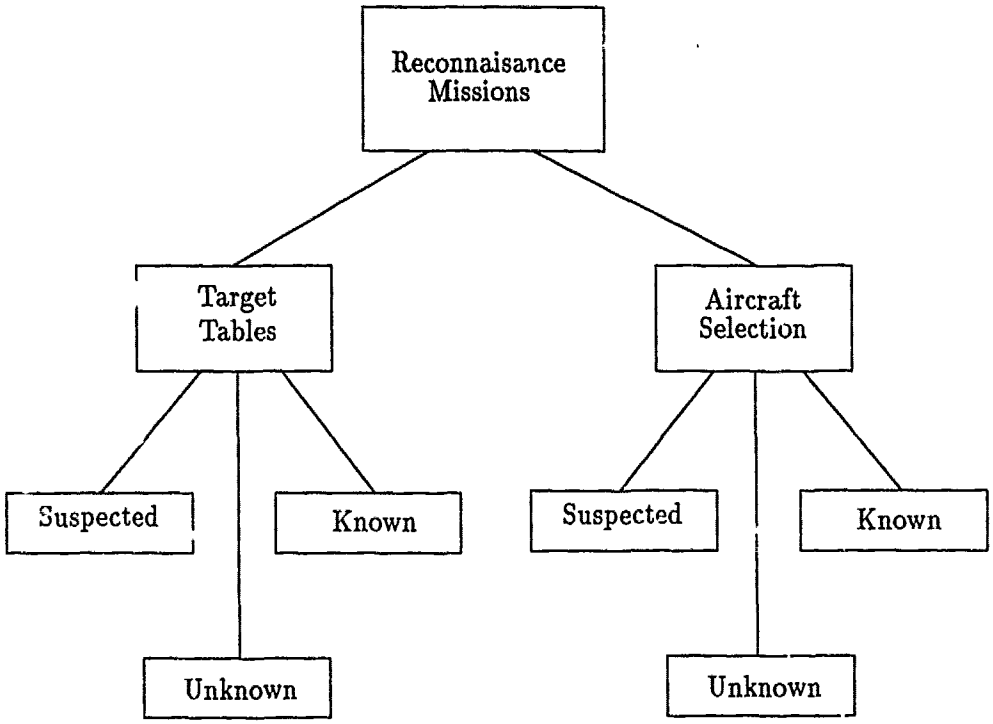

Figure 7. Reconnaissance Mission Design

The actual reconnaissance mission generation is done by target category as shown in Figure 7. The suspected targets are assigned first. Each suspected target has three three-ship packages flown against it as required by the written documentation (Air88). Then, two three-ship packages are assigned against each unknown target. Finally, one three-ship package is assigned against the targets in the known category.

After the reconnaissance missions have been assigned, the reconnaissance mission table is created. The table contains the type of aircraft, the quantity of aircraft and up to five targets on each line as shown in Table 16. The suspected targets show

Table 16. Reconnaissance Mission Table

AC Type	Quantity	Target 1	Target 2	Target 3	Target 4	Target 5
M25R	15	187	196	204	202	188
M25R	15	177	197	214	211	166
M25R	15	189	199	200	215	183
M25R	15	189	199	200	215	183
U17R	12	182	198	203	206	0
U17R	12	182	198	203	206	0
U17R	12	182	198	203	206	0

up in the table as three identical lines and the unknown targets as two identical lines while the known targets appear as single lines.

4.7 Summary

The design of the rule-base for the automated red player required the development of a target selection criterion. The criterion selected was in the form of an equation or a rating system which took into account the various resources of the possible targets for each type of mission. The design also requires that the aircraft data be available in a series of tables corresponding to the type of mission being planned. After this requirement is met, the generation of the mission packages can begin. The primary aircraft were allocated based upon their characteristics. The number of aircraft allocated for each target, depends on the weather conditions over the target, the target status and the aircraft's destructive index. Support aircraft are then allocated according to how well they match the needs of the mission package. After all the allocations are made the completed mission package was output in the appropriate mission table.

V. Conclusion and Recommendations

5.1 Conclusion

This thesis focused on the automation of the ATAF portion of the red player. Through the use of an expert system the goals, to reduce the time needed to plan and execute red moves and to develop a standardized, consistent red player, were realized. The paperwork required for planning the red player's actions during the ATAF portion of TWX are reduced to essentially zero. What was a four-hour, paper-intensive effort has been changed to a fifteen minute automated process. This frees the red players to concentrate on analyzing the blue players' decisions and gives the red players the time needed to prepare to help the student members of the blue teams understand the consequences of their decisions.

The second goal, a standardized red player, was also realized. The automated red player will handle similar situations according to the same set of rules and heuristics each time and for each group of blue players. This allows the success, or failure, of each seminar to be compared to the results obtained by other seminars separated by either distance or time. It also opens the door to in-depth comparison of different approaches taken by different blue teams allowing the teams to learn from others decisions. This automated red player is also portable to any computer system running the same expert system. The complete rule base can easily be stored on one double-sided, high-density, 5.25 inch floppy diskette. Even if a large number of rules are added in the future the rule base would continue to be extremely portable.

5.2 Recommendations for Further Work

At this point, the automated red player has been designed and a prototype has been implemented. The ATAF portion should be combined and used in tandem with the previous work. The combining of the two systems would further decrease the time needed to plan the red player's actions.

As the automated red player is refined and the Air Force Wargaming Center personnel become more familiar with the rule-based system, the restrictions which are built into the rules could be relaxed. In the manual system many rules were established to simplify the red player's work. With the automation of the red player, that is no longer a consideration. The restrictions on mission package composition can be relaxed to more accurately reflect reality.

A further recommendation, now that the red player is automated, is to consider the feasibility of using the same expert system shell to do preliminary analysis of each day's action by the blue player. This would provide valuable insights into the blue players' decisions on a daily basis. It would also capture the analysis expertise of the experienced red players, who are currently doing the analyses of the blue players' decisions, for use by future red players.

Appendix A. *User's Manual*

A.1 Introduction

The knowledge base developed as a result of this thesis was implemented under the Neuron Data expert system shell Nexpert Object on a Sun 386i minicomputer. Nexpert Object operates in the X Windows environment. This user's manual is divided into two parts. The first describes the tables used as input into the knowledge base and the tables that contain the results of the execution of the knowledge base. The second section explains how to run the knowledge base.

A.2 Table Format

The knowledge base for the automated red player retrieves information from flat tables in what is called the NXPDB (Nexpert database) format. Nexpert Object requires that the tables in the NXPDB format follow very stringent rules. The tables must be stored in files with an "NXP" extension and the file names must exactly match the names used in the knowledge base. The output tables are in the same format as the tables used for input and can be examined through the use of an editor, e.g., vi or ed.

An example of a table used by the knowledge base is shown below.

index	abid	status
1	65	.75
2	88	1.00
3	40	.50
4	74	0.00

The first line of the table contains the headers separated by vertical bars. Each field including the headers must end with a vertical bar "|" and the second line of the table between the headers and the data as well as the last line of the table must consist only of asterisks with no vertical bars and end in the same column as the last vertical bar in the header line. The number of spaces allocated to the headers or the data field, whichever is longer, determines the number of spaces Nexpert allocates to each field. The first field in each table must be an index field which contains a unique index for each record. The third though the next to the last lines contain the data. Each field must end with a vertical bar and the vertical bars must precisely line up with the corresponding vertical bars in the headers. No extra spaces or other characters are permitted after the last data field in a line. The output tables will be in the same format as the input tables.

The tables described above may either be built by a database query language, e.g., SQL, or manually. Regardless of how they are originally constructed, they may be manually modified at any time. The first record in a table, the first data line, will be accessed first and the other records will be accessed sequentially afterwards. For that reason it is important to place the higher priority targets and aircraft at the top of the table. If changes are made after the tables are built, the position of the new or changed record in the table will determine when it is processed.

A.3 Running the Knowledge Base

The knowledge base is run under Neuron Data's expert system shell Nexpert Object. It can be run on any computer system which can run Nexpert Object. The automated red player was developed and tested on a Sun 386i minicomputer. The knowledge base and the tables it uses must be in the same directory on the computer. The user should move to the directory which contains the knowledge base and enter Nexpert Object from there.

The procedure to run the automated red player are given here in a step by step format. A basic understanding of Nexpert Object is assumed. Once Nexpert Object is running, bring up the expert menu by clicking on the expert icon, the left middle icon in the Nexpert window. Load the knowledge base by selecting the "Load Knowledge Base" command in the expert menu and selecting "`autored.tkb`" when the knowledge base selection menu appears. After the knowledge base has been loaded, the various windows involved with the loading process will disappear. Next, select the "Restart Session" in the system menu to reinitialize all variables and objects in the knowledge base. Then select "Suggest..." from the system menu; a new window will open. Select "`autored.tkb`" in the new window; it will then appear under the heading "Suggest Keep". Use the mouse to click on "Ok & Knowcess"; this will start the processing of the automated red player. At this point a new window, the Session Window, will open and show some short comments about the condition of the run. When the execution is complete, the Session Window will display "End of Session".

Appendix B. *Rule-base Structure Overview*

This appendix contains a brief overview of the structure of the prototype rule-base for the automated red player in a series of figures. The top level overview, Figure 8, shows the structure of the rule-base presented in the following appendices. The ATAF box represents the top level controller which activates the tasks in the proper sequence. The boxes below it represent the various tasks in the rule-base.

Figure 9 shows the structure of the long range offensive counter air rule-base in the prototype. The boxes under the lrMapping box represent the various hypotheses or goals that must be met to continue processing the packaging task. The processing of the goals starts at the left and moves to the right but certain conditions may cause the rule-base to backtrack and return to a goal which had been processed previously.

The defense suppression task structure is shown in Figure 10. It is a very simple, straightforward process of selecting the aircraft from the tables with few restrictions.

Since all three reconnaissance tasks are structured alike only Figure 11 is presented showing the structure of all three tasks. It is essentially the same structure as the long range OCA portion.

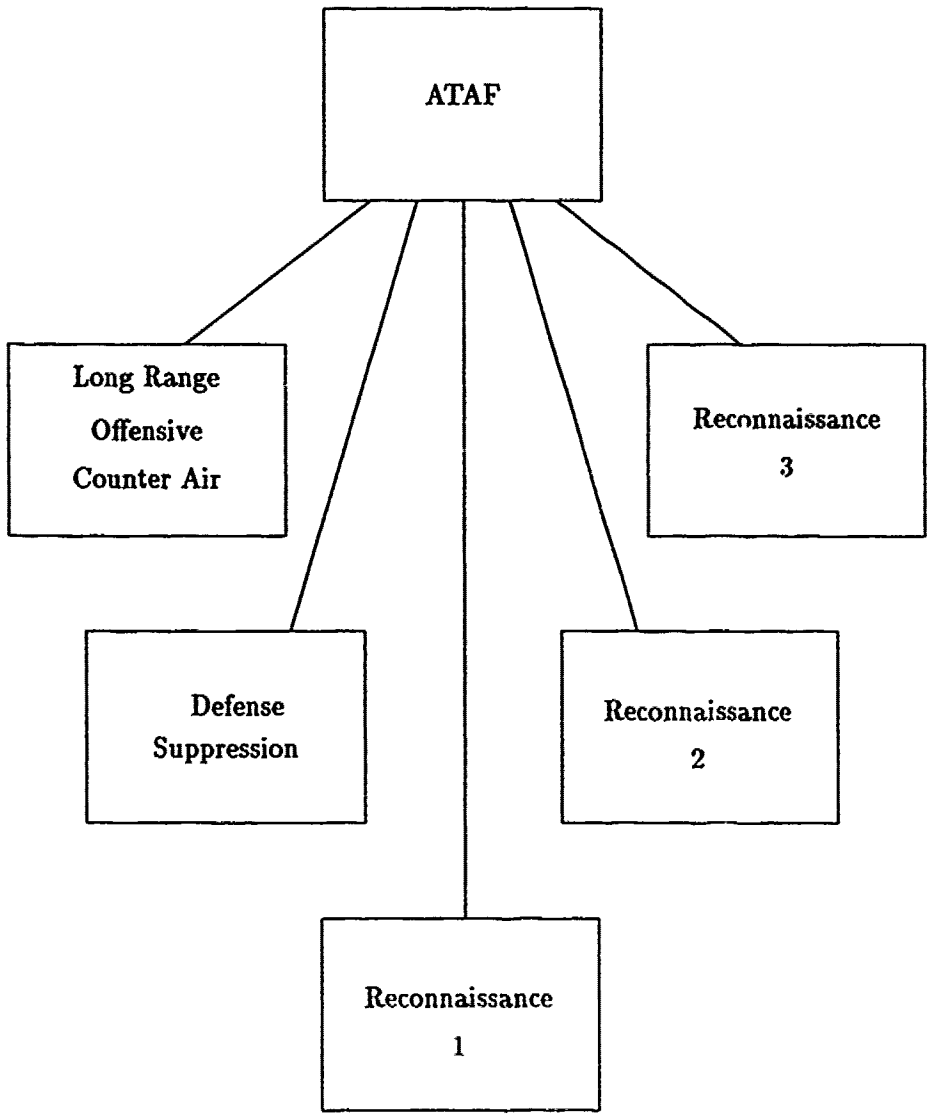

Figure 8. ATAF Top Level Structure

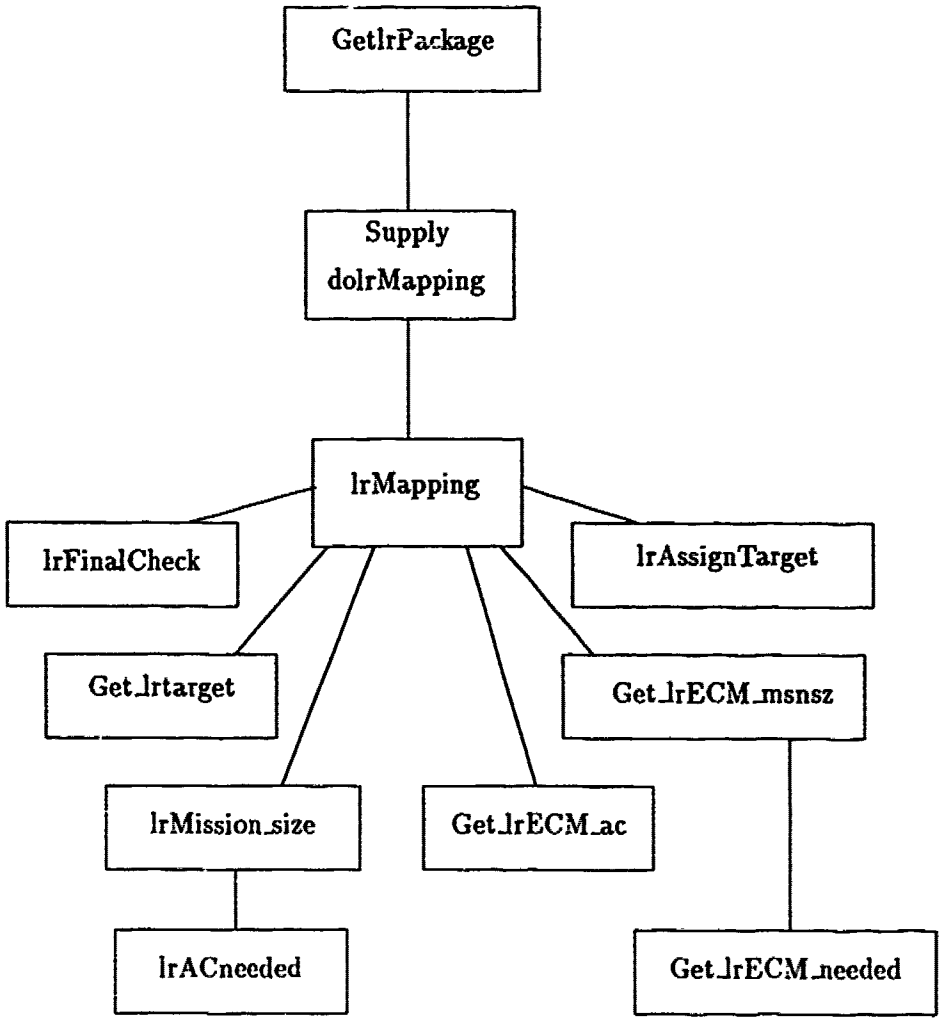

Figure 9. Long Range OCA Structure

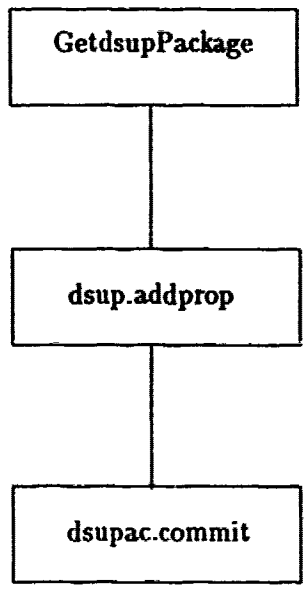

Figure 10. Defense Suppression Task Structure

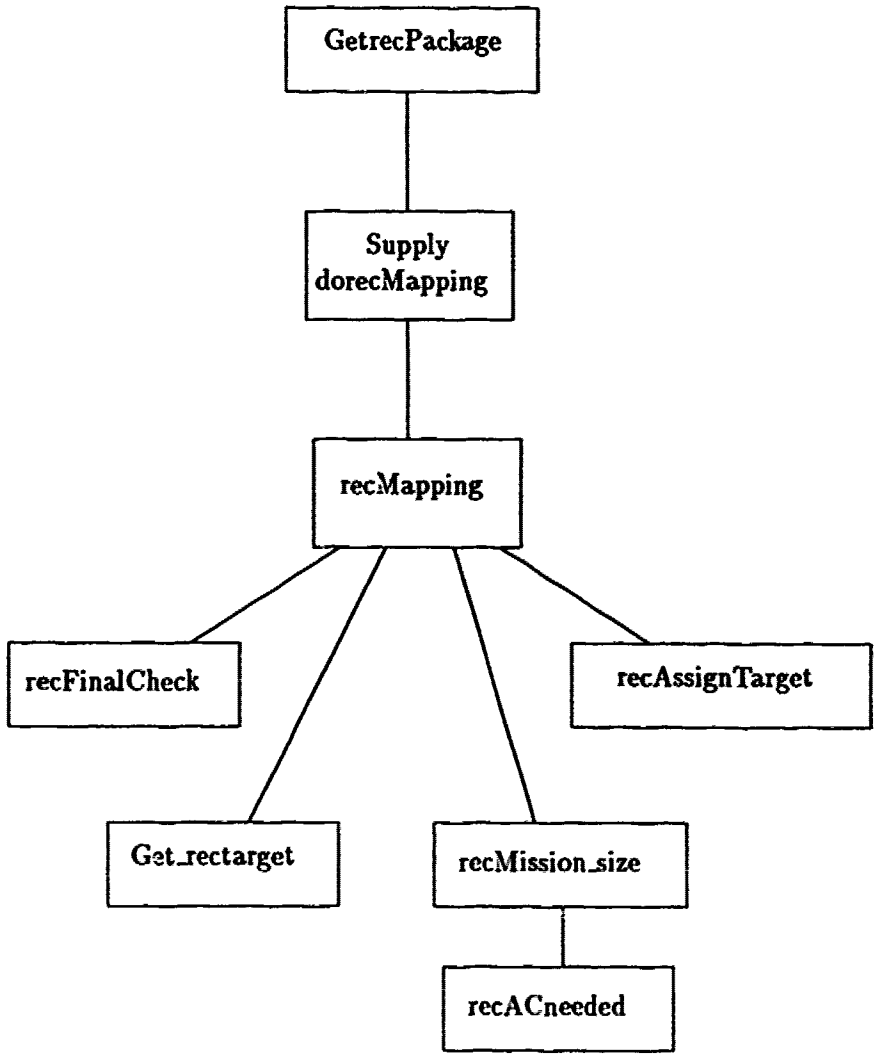

Figure 11. Reconnaissance Tasks Structure

Appendix C. *Rules*

C.1 Introduction

This appendix contains the rule-base for the automated red player. The rules are presented in the format Nexpert Object outputs them when requested to "Write to File" in the rule editor.

The rules are presented in sections within the rule-base. Variable names and hypotheses belonging to the different tasks can be identified by abbreviations contained in the names. Variables and hypotheses pertaining to the long range offensive counter air task contain "lr" or "lroca." Those used in the defense suppression task contain "dsup." The recconnaisance task items contain "rec1," "rec2," "rec3," "recce1," "recce2," or "recce3,"

The conditions and actions are the same as described in Chapter 4. The hypothesis of a rule can be considered to be the goal of the rule and several rules may have the same hypothesis. The inference category determines the order of execution among rules with the same hypotheses; it is set to one (1) as the default. The name is optional as are the comments.

C.2 The Automated Red Player Rule-base

```
CONDITIONS :
    Name        GetlrPackage.Hypo GetlrPackage.Hypo
    Name        dsup_missions dsup_missions
    Name        Getrec1Package.Hypo Getrec1Package.Hypo
    Name        rec2GetPackage.Hypo rec2GetPackage.Hypo
    Name        Getrec3Package.Hypo Getrec3Package.Hypo
HYPOTHESIS :    ataf.Hypo
       INFERENCE CATEGORY :    1
NAME :
       Comments : ATAF automated red player prototype by Karl Kabanek
```

```
CONDITIONS :
    Retrieve    "dsupac.nxp"
      @TYPE=NXPDB;@FILL=ADD;@NAME="'obj_'!index!";@CREATE=|dsupac|;
      @PROPS=type,qnty;@FIELDS="type","qnty";
    Name        <|dsupac|>.addprop <|dsupac|>.addprop
    >           LENGTH(<|msac|>) 0
HYPOTHESIS :    dsup_missions
ACTIONS :
    Write       "dsupms.nxp"
      @TYPE=NXPDB;@FILL=NEW;@NAME="'MS_'!index(10)!";@PROPS=type,qnty;
      @FIELDS="type(10)","qnty(10)";@ATOMS=<|msac|>;
    Write       "dsup_2.nxp"
      @TYPE=NXPDB;@FILL=NEW;@NAME="'ACR_'!index(10)!";@PROPS=type,qnty;
      @FIELDS="type(10)","qnty(10)";@ATOMS=<|acr|>;
        INFERENCE CATEGORY :    1
NAME :  Top_level_dsup

CONDITIONS :
    >           dsupqnty 0
    Name        dspglobal.qnty/sortsz*sortsz dspglobal.sortqnty
    Name        dspglobal.qnty-dspglobal.sortqnty dspglobal.rmdr
    <           dspglobal.sortqnty-dsupqnty 0
HYPOTHESIS :    dsupac_commit
ACTIONS :
    Do          dsupqnty-dspglobal.sortqnty dsupqnty
    Do          |DSPWMEM|.index+1 |DSPWMEM|.index
    CreateObjec 'MS_'\|DSPWMEM|.index\ |msac|
    Do          dspglobal.type 'MS_'\|DSPWMEM|.index\.type
    Do          dspglobal.sortqnty 'MS_'\|DSPWMEM|.index\.qnty
    Do          dspglobal.rmdr dspglobal.qnty
    CreateObjec 'ACR_'\|DSPWMEM|.index\ |acr|
    Do          dspglobal.type 'ACR_'\|DSPWMEM|.index\.type
    Do          dspglobal.qnty 'ACR_'\|DSPWMEM|.index\.qnty
        INFERENCE CATEGORY :    50
NAME :  select_dsupac_2
```

```
CONDITIONS :
    >           dsupqnty 0
    Name        dspglobal.qnty/sortsz*sortsz dspglobal.sortqnty
    Name        dspglobal.qnty-dspglobal.sortqnty dspglobal.rmdr
    >=          dspglobal.sortqnty-dsupqnty 0
HYPOTHESIS :    dsupac_commit
ACTIONS :
    Do          dspglob..sortqnty-dsupqnty+dspglobal.rmdr dspglobal.qnty
    Do          |DSPWMEM|.index+1 |DSPWMEM|.index
    CreateObjec 'MS_'\|DSPWMEM|.index\ |msac|
    Do          dspglobal.type 'MS_'\|DSPWMEM|.index\.type
    Do          dsupqnty 'MS_'\|DSPWMEM|.index\.qnty
    Do          0 dsupqnty
    CreateObjec 'ACR_'\|DSPWMEM|.index\ |acr|
    Do          dspglobal.type 'ACR_'\|DSPWMEM|.index\.type
    Do          dspglobal.qnty 'ACR_'\|DSPWMEM|.index\.qnty
        INFERENCE CATEGORY :    100
NAME :  select_dsupac_1

CONDITIONS :
    =           dsupqnty 0
    NotEqual    dspglobal.type 'ACR_'\|DSPWMEM|.index\.type
HYPOTHESIS :    dsupac_commit
ACTIONS :
    Do          |DSPWMEM|.index+1 |DSPWMEM|.index
    CreateObjec 'ACR_'\|DSPWMEM|.index\ |acr|
    Do          dspglobal.type 'ACR_'\|DSPWMEM|.index\.type
    Do          dspglobal.qnty 'ACR_'\|DSPWMEM|.index\.qnty
        INFERENCE CATEGORY :    200
NAME :  select_dsupac_0
```

```
CONDITIONS :
    =           LRWMEM.ecmtot 0
    =           curlrMap.actot 0
HYPOTHESIS :    Get_lrECM_ac.Hypo
ACTIONS :
    Do          LRWMEM.ecmindx+1 LRWMEM.ecmindx
    CreateObjec 'lrECM_'\LRWMEM.ecmindx\ |lrECMac|
    Do          LRWMEM.ecmtyp 'lrECM_'\LRWMEM.ecmindx\.ecmtyp
    Do          LRWMEM.ecmtot 'lrECM_'\LRWMEM.ecmindx\.ecmtot
    Retrieve    "lrecm.nxp"
      @TYPE=NXPDB;@SLOTS=LRWMEM.ecmtyp,LRWMEM.ecmtot;@FIELDS="type","qnty";
      @CURSOR=LRWMEM.ecmCursor;
    Do          LRWMEM.ecmtyp curlrMap.ecmtyp
        INFERENCE CATEGORY :    50
NAME :
        Comments : Out of ECM ac and strike ac

CONDITIONS :
    Is          LRWMEM.ecmtyp UNKNOWN
HYPOTHESIS :    Get_lrECM_ac.Hypo
ACTIONS :
    Retrieve    "lrecm.nxp"
      @TYPE=NXPDB;@SLOTS=LRWMEM.ecmtyp,LRWMEM.ecmtot;@FIELDS="type","qnty";
      @CURSOR=LRWMEM.ecmCursor;
    Do          LRWMEM.ecmtyp curlrMap.ecmtyp
        INFERENCE CATEGORY :    150
NAME :  Start_lrECM_table_use
        Comments : Get the first entry in the ecm table
```

```
CONDITIONS :
    =           LRWMEM.ecmtot 0
    <>          curlrMap.actot 0
HYPOTHESIS :    Get_lrECM_ac.Hypo
ACTIONS :
    Do          LRWMEM.ecmindx+1 LRWMEM.ecmindx
    CreateObjec 'lrECM_'\LRWMEM.ecmindx\ |lrECMac|
    Do          LRWMEM.ecmtyp 'lrECM_'\LRWMEM.ecmindx\.ecmtyp
    Do          LRWMEM.ecmtot 'lrECM_'\LRWMEM.ecmindx\.ecmtot
    Do          TRUE curlrMap.doAssign
    Retrieve    "lrecm.nxp"
      @TYPE=NXPDB;@SLOTS=LRWMEM.ecmtyp,LRWMEM.ecmtot;@FIELDS="type","qnty";
      @CURSOR=LRWMEM.ecmCursor;
    Do          LRWMEM.ecmtyp curlrMap.ecmtyp
       INFERENCE CATEGORY :    75
NAME :  Retrieve_ecm_ac_from_table
        Comments : This gets the next entry in the ecm table when the previous
         entry run out of ac.

CONDITIONS :
    >           LRWMEM.ecmtot 0
HYPOTHESIS :    Get_lrECM_ac.Hypo
       INFERENCE CATEGORY :    100
NAME :  Get_ecm_ac_from_sa e_entr
        Comments : This continues   .  .e the same type of ecm ac as before

CONDITIONS :
    Name        Get_lrECM_needed.Hypo Get_lrECM_needed.Hypo
    >=          LRWMEM.ecmtot-LRWMEM.ecmneeded 0
HYPOTHESIS :    Get_lrECM_msnsz.Hypo
ACTIONS :
    Do          LRWMEM.ecmneeded LRWMEM.ecmmsnsz
       INFERENCE CATEGORY :    200
NAME :  Sufficient_lrECM_msnsz
        Comments : Sufficient ECM ac for this mission
```

```
CONDITIONS :
    Name        Get_lrECM_needed.Hypo Get_lrECM_needed.Hypo
    <           LRWMEM.ecmtot-LRWMEM.ecmneeded 0
HYPOTHESIS :    Get_lrECM_msnsz.Hypo
ACTIONS :
    Do          0 LRWMEM.msnsz
    Do          0 LRWMEM.ecmmsnsz
        INFERENCE CATEGORY :    150
NAME :
        Comments : No more ECM ac available of this type

CONDITIONS :
    >           LRWMEM.msnsz 0
HYPOTHESIS :    Get_lrECM_needed.Hypo
ACTIONS :
    Do          1 LRWMEM.ecmneeded
        INFERENCE CATEGORY :    100
NAME :  Standard_lrECM_msnsz
        Comments : ECM needed for regular mission

CONDITIONS :
    <=          LRWMEM.msnsz 0
HYPOTHESIS :    Get_lrECM_needed.Hypo
ACTIONS :
    Do          0 LRWMEM.ecmneeded
        INFERENCE CATEGORY :    50
NAME :  Non_mission_lrECM_msnsz
        Comments : No ecm ac for a non-mission.

CONDITIONS :
    <>          lrHold.target 0
    Name        lrHold.target LRWMEM.curTarget
    Name        lrHold.status LRWMEM.curstatus
HYPOTHESIS :    Get_lrtarget.Hypo
ACTIONS :
    Reset       lrHold.target
    Reset       lrHold.status
        INFERENCE CATEGORY :    25
NAME :
```

```
CONDITIONS :
    =           lrHold.target 0
    Retrieve    "lrtgt.nxp"
      @TYPE=NXPDB;@SLOTS=LRWMEM.curTarget,LRWMEM.curstatus;
      @FIELDS="abid","status";@CURSOR=LRWMEM.targCursor;
HYPOTHESIS :    Get_lrtarget.Hypo
       INFERENCE CATEGORY :    100
NAME :  Find_the_target

CONDITIONS :
    Retrieve    "lrac.nxp"
      @TYPE=NXPDB;@FILL=ADD;@NAME="'AC_'!Indx!";@CREATE=|lrSupply|;
      @PROPS=actyp,actot,didg,didb,didp,ding,dinp,dinb;
      @FIELDS="type","qnty","didg","didb","didp","ding","dinp","dinb";
    Name        <|lrSupply|>.dolrMapping <|lrSupply|>.dolrMapping
    >           LENGTH(<|lrAssignments|>) 0
HYPOTHESIS :    GetlrPackage.Hypo
ACTIONS :
    Write       "lrms.nxp"
      @TYPE=NXPDB;@FILL=NEW;@NAME="'Assign_'!Indx(8)!";
      @PROPS=actyp,actot,tgt1,sort1,tgt2,sort2,tgt3,sort3,tgt4,sort4,
          tgt5,sort5,ecmtyp,ecmtot;
      @FIELDS="actyp(10)","qnty(8)","tgt1(7)","sort1(5)","tgt2(7)","sort2(5)",
          "tgt3(7)","sort3(5)","tgt4(7)","sort4(5)","tgt5(7)","sort5(5)",
          "ecmtyp(7)","ecmtot(7)";@ATOMS=<|lrAssignments|>;
    Write       "lrac_2.nxp" @TYPE=NXPDB;@FILL=NEW;@NAME="'AC_'!Indx(8)!";
      @PROPS=actyp,actot,didg,didp,didb,ding,dinp,dinb;
      @FIELDS="actyp(10)","qnty(8)","didg(5)","didp(5)","didb(5)",
          "ding(5)","dinp(5)","dinb(5)";@ATOMS=<|lrSupply|>;
    Do          LRWMEM.ecmindx+1 LRWMEM.ecmindx
    CreateObjec 'lrECM_'\LRWMEM.ecmindx\ |lrECMac|
    Do          LRWMEM.ecmtyp 'lrECM_'\LRWMEM.ecmindx\.ecmtyp
    Do          LRWMEM.ecmtot 'lrECM_'\LRWMEM.ecmindx\.ecmtot
    Do          lrECM table.Hypo lrECM_table.Hypo
    Write       "lrecm_2.nxp" @TYPE=NXPDB;@FILL=NEW;@NAME="'lrECM_'!indx!";
      @PROPS=ecmtyp,ecmtot;@FIELDS="type","qnty";@ATOMS=<|lrECMac|>;
       INFERENCE CATEGORY :    1
NAME :  Top_Level_Package_Control
       Comments : This rule represents the top level control structure:
       Build the table of planes to be assigned.  Assign planes (map them on)
       to available targets.  If any assignments have been made, update the
       database tables.
```

```
CONDITIONS :
    Retrieve    "rec1ac.nxp"
      @TYPE=NXPDB;@FILL=ADD;@NAME="'AC_'!Indx!";@CREATE=|rec1Supply|;
      @PROPS=actyp,actot;@FIELDS="actyp","qnty";
    Name        <|rec1Supply|>.dorec1Mapping <|rec1Supply|>.dorec1Mapping
    >           LENGTH(<|rec1Assignments|>) 0
HYPOTHESIS :    Getrec1Package.Hypo
ACTIONS :
    Write       "rec1ms.nxp"
     @TYPE=NXPDB;@FILL=NEW;@NAME="'Assign_'!Indx(8)!";
     @PROPS=actyp,actot,tgt1,tgt2,tgt3,tgt4,tgt5;
     @FIELDS="actyp(10)","qnty(8)","tgt1(7)","tgt2(7)","tgt3(7)",
      "tgt4(7)","tgt5(7)";@ATOMS=<|rec1Assignments|>;
    Write       "rec2ac.nxp"
      @TYPE=NXPDB;@FILL=NEW;@NAME="'AC_'!Indx(8)!";@PROPS=actyp,actot;
      @FIELDS="actyp(10)","qnty(8)";@ATOMS=<|rec1Supply|>;
        INFERENCE CATEGORY :    1
NAME :  Top_Level_Package_Control
        Comments : This rule represents the top level control structure: Build
         the table of planes to be assigned.  Assign planes (map them on) to
         available targets.  If any assignments have been made, update the
         database tables.
```

```
CONDITIONS :
    Retrieve    "rec3ac.nxp"
     @TYPE=NXPDB;@FILL=ADD;@NAME="'AC_'!Indx!";@CREATE=|rec3Supply|;
     @PROPS=actyp,actot;@FIELDS="actyp","qnty";
    Name        <|rec3Supply|>.dorec3Mapping <|rec3Supply|>.dorec3Mapping
    >           LENGTH(<|rec3Assignments|>) 0
HYPOTHESIS :    Getrec3Package.Hypo
ACTIONS :
    Write       "rec3ms.nxp"
      @TYPE=NXPDB;@FILL=NEW;@NAME="'Assign_'!Indx(8)!";
      @PROPS=actyp,actot,tgt1,tgt2,tgt3,tgt4,tgt5;
      @FIELDS="actyp(10)","qnty(8)","tgt1(7)","tgt2(7)","tgt3(7)",
           "tgt4(7)","tgt5(7)";@ATOMS=<|rec3Assignments|>;
    Write       "recaclft.nxp"
      @TYPE=NXPDB;@FILL=NEW;@NAME="'AC_'!Indx(8)!";@PROPS=actyp,actot;
      @FIELDS="actyp(10)","qnty(8)";@ATOMS=<|rec3Supply|>;
        INFERENCE CATEGORY :    1
NAME :  Top_Level_Package_Control
        Comments : This rule represents the top level control structure: Build
         the table of planes to be assigned.  Assign planes (map them on) to
         available targets.  If any assignments have been made, update the
         database tables.
```

```
CONDITIONS :
    >=          LRWMEM.curstatus .625
    <           LRWMEM.curstatus .875
    Name        MAX(LRWMEM.didg,0.7) LRWMEM.effsrt
    Name        CEIL(25/LRWMEM.effsrt) LRWMEM.acneeded
HYPOTHESIS :    lrACneeded.Hypo
        INFERENCE CATEGORY :    75
NAME :  Find_AC_needed_75
        Comments : This rule determines the mission size needed for a
         target at 75

CONDITIONS :
    >=          LRWMEM.curstatus .375
    <           LRWMEM.curstatus .625
    Name        MAX(LRWMEM.didg,0.7) LRWMEM.effsrt
    Name        CEIL(15/LRWMEM.effsrt) LRWMEM.acneeded
HYPOTHESIS :    lrACneeded.Hypo
        INFERENCE CATEGORY :    50
NAME :  Find_AC_needed_50
        Comments : This rule find the mission size needed for a
         target with a status of 50

CONDITIONS :
    >           LRWMEM.curstatus 0
    <           LRWMEM.curstatus .375
    Name        MAX(LRWMEM.didg,0.7) LRWMEM.effsrt
    Name        CEIL(5/LRWMEM.effsrt) LRWMEM.acneeded
HYPOTHESIS :    lrACneeded.Hypo
        INFERENCE CATEGORY :    25
NAME :  Find_AC_needed_25
        Comments : This rule determines the mission size needed for
         targets with a status of 25

CONDITIONS :
    >=          LRWMEM.curstatus .875
    Name        MAX(LRWMEM.didg,0.7) LRWMEM.effsrt
    Name        CEIL(35/LRWMEM.effsrt) LRWMEM.acneeded
HYPOTHESIS :    lrACneeded.Hypo
        INFERENCE CATEGORY :    100
NAME :  Find_AC_needed_100
        Comments : This rule determines the mission size needed for
         targets with a status of 100
```

```
CONDITIONS :
    >           LRWMEM.msnsz 0
    IsNot       curlrMap.tgt5 KNOWN
    Name        LRWMEM.curTarget curlrMap.tgt5
    Name        curlrMap.actot+LRWMEM.msnsz curlrMap.actot
    Name        LRWMEM.msnsz curlrMap.sort5
    Name        curlrMap.ecmtot+LRWMEM.ecmmsnsz curlrMap.ecmtot
    Name        TRUE curlrMap.doAssign
HYPOTHESIS :    lrAssignTarget.Hypo
ACTIONS :
    Do          TRUE LRWMEM.retflag
        INFERENCE CATEGORY :    60
NAME :  Target_Assignment

CONDITIONS :
    >           LRWMEM.msnsz 0
    IsNot       curlrMap.tgt1 KNOWN
    Name        LRWMEM.curTarget curlrMap.tgt1
    Name        curlrMap.actot+LRWMEM.msnsz curlrMap.actot
    Name        LRWMEM.msnsz curlrMap.sort1
    Name        curlrMap.ecmtot+LRWMEM.ecmmsnsz curlrMap.ecmtot
HYPOTHESIS :    lrAssignTarget.Hypo
ACTIONS :
    Do          FALSE LRWMEM.retflag
        INFERENCE CATEGORY :    100
NAME :  Target_Assignment
        Comments : If the first target is being assigned, copy the
         target value and update the aircraft total.

CONDITIONS :
    >           LRWMEM.msnsz 0
    IsNot       curlrMap.tgt3 KNOWN
    Name        LRWMEM.curTarget curlrMap.tgt3
    Name        curlrMap.actot+LRWMEM.msnsz curlrMap.actot
    Name        LRWMEM.msnsz curlrMap.sort3
    Name        curlrMap.ecmtot+LRWMEM.ecmmsnsz curlrMap.ecmtot
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    80
NAME :  Target_Assignment
        Comments : If a third target is being assigned, copy the target
         value and update the aircraft total.
```

```
CONDITIONS :
    >           LRWMEM.msnsz 0
    IsNot       curlrMap.tgt4 KNOWN
    Name        LRWMEM.curTarget curlrMap.tgt4
    Name        curlrMap.actot+LRWMEM.msnsz curlrMap.actot
    Name        LRWMEM.msnsz curlrMap.sort4
    Name        curlrMap.ecmtot+LRWMEM.ecmmsnsz curlrMap.ecmtot
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    70
NAME :  Target_Assignment
        Comments : If a fourth target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    >           LRWMEM.msnsz 0
    IsNot       curlrMap.tgt2 KNOWN
    Name        LRWMEM.curTarget curlrMap.tgt2
    Name        curlrMap.actot+LRWMEM.msnsz curlrMap.actot
    Name        LRWMEM.msnsz curlrMap.sort2
    Name        curlrMap.ecmtot+LRWMEM.ecmmsnsz curlrMap.ecmtot
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    90
NAME :  Target_Assignment
        Comments : If a second target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    =           LRWMEM.msnsz 0
    =           lrHold.target 0
    IsNot       curlrMap.tgt5 KNOWN
    Name        LRWMEM.curTarget lrHold.target
    Name        LRWMEM.curstatus lrHold.status
    Name        0 curlrMap.tgt5
    Name        0 curlrMap.sort5
HYPOTHESIS :    lrAssignTarget.Hypo
ACTIONS :
    Do          TRUE LRWMEM.retflag
        INFERENCE CATEGORY :    57
NAME :  Target_5_assignment
        Comments : This rule fills the target and sortie fields with 0s when
          there are no ac left.
```

```
CONDITIONS :
    =           LRWMEM.msnsz 0
    <>          lrHold.target 0
    IsNot       curlrMap.tgt5 KNOWN
    Name        0 curlrMap.tgt5
    Name        0 curlrMap.sort5
HYPOTHESIS :    lrAssignTarget.Hypo
ACTIONS :
    Do          TRUE LRWMEM.retflag
        INFERENCE CATEGORY :    53
NAME :  Target_5_assignment
        Comments : This rule fills the sortie and target fields with 0s when
          there are no more ac

CONDITIONS :
    =           LRWMEM.msnsz 0
    <>          lrHold.target 0
    IsNot       curlrMap.tgt4 KNOWN
    Name        0 curlrMap.tgt4
    Name        0 curlrMap.sort4
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    63
NAME :  Target_4_assignment
        Comments : This rule fills the sortier and target fields with 0s
         when there are no more ac.

CONDITIONS :
    =           LRWMEM.msnsz 0
    =           lrHold.target 0
    IsNot       curlrMap.tgt4 KNOWN
    Name        LRWMEM.curTarget lrHold.target
    Name        LRWMEM.curstatus lrHold.status
    Name        0 curlrMap.tgt4
    Name        0 curlrMap.sort4
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    67
NAME :  Target_4_assignment
        Comments : This rule fills the target and sortie fields with 0s
         when there are  no ac left
```

```
CONDITIONS :
    =           LRWMEM.msnsz 0
    <>          lrHold.target 0
    IsNot       curlrMap.tgt3 KNOWN
    Name        0 curlrMap.tgt3
    Name        0 curlrMap.sort3
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    73
NAME :  Target_3_assignment
        Comments : This rule fills the target and sortie fields with 0s when
          there are no more ac.

CONDITIONS :
    =           LRWMEM.msnsz 0
    =           lrHold.target 0
    IsNot       curlrMap.tgt3 KNOWN
    Name        LRWMEM.curTarget lrHold.target
    Name        LRWMEM.curstatus lrHold.status
    Name        0 curlrMap.tgt3
    Name        0 curlrMap.sort3
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    77
NAME :  Target_3_assignment
        Comments : This fills the target and sortie fields with 0s when
          there are no ac left

CONDITIONS :
    =           LRWMEM.msnsz 0
    <>          lrHold.target 0
    IsNot       curlrMap.tgt2 KNOWN
    Name        0 curlrMap.tgt2
    Name        0 curlrMap.sort2
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    83
NAME :  Target_2_assignment
        Comments : This rule fills the sortie and target fields with 0s when
          there are no more ac.
```

```
CONDITIONS :
    =           LRWMEM.msnsz 0
    =           lrHold.target 0
    IsNot       curlrMap.tgt2 KNOWN
    Name        LRWMEM.curTarget lrHold.target
    Name        LRWMEM.curstatus lrHold.status
    Name        0 curlrMap.tgt2
    Name        0 curlrMap.sort2
HYPOTHESIS :    lrAssignTarget.Hypo
        INFERENCE CATEGORY :    87
NAME :  Target_2_assignment
        Comments : This rule fills the target abd sortie fields with 0s when
          there are no more ac.

CONDITIONS :
    =           LRWMEM.msnsz 0
    <>          lrHold.target 0
    IsNot       curlrMap.tgt1 KNOWN
    Name        0 curlrMap.tgt1
    Name        0 curlrMap.sort1
HYPOTHESIS :    lrAssignTarget.Hypo
ACTIONS :
    Do          FALSE LRWMEM.retflag
        INFERENCE CATEGORY :    93
NAME :  Target_1_assignment
        Comments : This rule fills the sortie and target fields with 0s when
         there are no more ac

CONDITIONS :
    =           LRWMEM.msnsz 0
    =           lrHold.target 0
    IsNot       curlrMap.tgt1 KNOWN
HYPOTHESIS :    lrAssignTarget.Hypo
ACTIONS :
    Do          LRWMEM.curTarget lrHold.target
    Do          LRWMEM.curstatus lrHold.status
    Do          0 LRWMEM.curTarget
    Do          0 curlrMap.sort1
    Do          FALSE LRWMEM.retflag
        INFERENCE CATEGORY :    97
NAME :  Null_target_1
```

```
CONDITIONS :
    <>          LRWMEM.ecmCursor -1
    Retrieve    "lrecm.nxp" @TYPE=MXPDB;@SLOTS=LRWMEM.ecmtyp,LRWMEM.ecmtot;
      @FIELDS="type","qnty";@CURSOR=LRWMEM.ecmCursor;
    <>          LRWMEM.ecmCursor -1
HYPOTHESIS :    lrECM_table.Hypo
ACTIONS :
    Do          LRWMEM.ecmindx+1 LRWMEM.ecmindx
    CreateObjec 'lrECM_'\LRWMEM.ecmindx\ |lrECMac|
    Do          LRWMEM.ecmtyp 'lrECM_'\LRWMEM.ecmindx\.ecmtyp
    Do          LRWMEM.ecmtot 'lrECM_'\LRWMEM.ecmindx\.ecmtot
    Do          lrECM_table.Hypo lrECM_table.Hypo
    Reset       lrECM_table.Hypo
        INFERENCE CATEGORY :    1
NAME :

CONDITIONS :
    Is          LRWMEM.targCursor KNOWN
    =           LRWMEM.targCursor -1
    No          <|lrSupply|>.Processed
    Name        TRUE <|lrSupply|>.dolrMapping
HYPOTHESIS :    lrFinalCheck.Hypo
        INFERENCE CATEGORY :    1
NAME :  Stop_Assignment_lrMappings
        Comments : If LRWMEM.targCursor is equal to -1 then the end of the
         table has been reached and there are no more targets to be assigned.

CONDITIONS :
    Reset       lrFinalCheck.Hypo
    Name        lrFinalCheck.Hypo lrFinalCheck.Hypo
    >           curlrMap.actot 0
    Name        TRUE curlrMap.doAssign
HYPOTHESIS :    lrMapping.Hypo
        INFERENCE CATEGORY :    50
NAME :  Stop_Assignment_Loop
        Comments : Since lrMapping.Hypo will not be reset, this rule will
         always terminate the loop.  It is only evaluated when all possible
         assignments from the 'AC_XX' object have been done.  If 'curlrMap' is
         partially filled it triggers creation of a new assignment objec
```

```
CONDITIONS :
    >           LRWMEM.actot 3
    Name        Get_lrtarget.Hypo Get_lrtarget.Hypo
    <>          LRWMEM.targCursor -1
    Name        lrMission_size.Hypo lrMission_size.Hypo
    Reset       Get_lrECM_ac.Hypo
    Name        Get_lrECM_ac.Hypo Get_lrECM_ac.Hypo
    <>          LRWMEM.ecmCursor -1
    Name        Get_lrECM_msnsz.Hypo Get_lrECM_msnsz.Hypo
    Name        lrAssignTarget.Hypo lrAssignTarget.Hypo
    Name        LRWMEM.actot-LRWMEM.msnsz LRWMEM.actot
    Name        LRWMEM.ecmtot-LRWMEM.ecmmsnsz LRWMEM.ecmtot
HYPOTHESIS :    lrMapping.Hypo
ACTIONS :
    Reset       lrMapping.Hypo
    Reset       lrMission_size.Hypo
    Reset       lrACneeded.Hypo
    Reset       Get_lrECM_msnsz.Hypo
        INFERENCE CATEGORY :    100
NAME :  Define_Assignment_Loop
        Comments : This rule will loop until there are no available planes
         from the current 'AC_XX' object.

CONDITIONS :
    Name        lrACneeded.Hypo lrACneeded.Hypo
    <>          LRWMEM.targCursor -1
    <           LRWMEM.actot-0.5*LRWMEM.acneeded 0
    <           LRWMEM.actot 4
HYPOTHESIS :    lrMission_size.Hypo
ACTIONS :
    Do          0 LRWMEM.msnsz
        INFERENCE CATEGORY :    150
NAME :  Determine_the_mission_size_when_there_are_essentially_no_ac_left
        Comments : This rule sets the mission size to 0 when there are too
         few ac left to use and allow the next type of ac to be used.
```

```
CONDITIONS :
    Name          lrACneeded.Hypo lrACneeded.Hypo
    <>            LRWMEM.targCursor -1
    <             LRWMEM.actot-0.5*LRWMEM.acneeded 0
    >=            LRWMEM.actot 4
HYPOTHESIS :      lrMission_size.Hypo
ACTIONS :
    Do            LRWMEM.actot LRWMEM.msnsz
        INFERENCE CATEGORY :    175
NAME :  Determine_mission_size_when_there_are_minimal_ac
        Comments : This rule determines the mission size when there are few
         ac of a particular type left.

CONDITIONS :
    Name          lrACneeded.Hypo lrACneeded.Hypo
    >=            LRWMEM.actot-LRWMEM.acneeded 0
HYPOTHESIS :      lrMission_size.Hypo
ACTIONS :
    Do            LRWMEM.acneeded LRWMEM.msnsz
        INFERENCE CATEGORY :    200
NAME :  Determine_mission_size_when_there_are_ample_ac
        Comments : This rule determines the mission size when there are more
          ac available.

CONDITIONS :
    Name          lrACneeded.Hypo lrACneeded.Hypo
    <>            LRWMEM.targCursor -1
    >=            LRWMEM.actot-0.5*LRWMEM.acneeded 0
HYPOTHESIS :      lrMission_size.Hypo
ACTIONS :
    Do            LRWMEM.actot LRWMEM.msnsz
        INFERENCE CATEGORY :    125
NAME :  Determine_mission_size_for_target
        Comments : This rule determines the mission size for a target in the
         table when the number of available ac is less than the amount needed.
```

```
CONDITIONS :
    Name        lrACneeded.Hypo lrACneeded.Hypo
    =           LRWMEM.targCursor -1
    <=          LRWMEM.actot-LRWMEM.acneeded 0
HYPOTHESIS :    lrMission_size.Hypo
ACTIONS :
    Do          LRWMEM.actot LRWMEM.msnsz
        INFERENCE CATEGORY :    100
NAME :  Determine_mission_size_for_last_target
        Comments : This rule determines the size of the mission for the last
         target in the list when the number of ac left is < or =  the
         number of ac needed.

CONDITIONS :
    IsNot       rec1curMap.tgt3 KNOWN
    Name        REC1WMEM.curTarget rec1curMap.tgt3
    Name        rec1curMap.actot+3 rec1curMap.actot
HYPOTHESIS :    rec1AssignTarget.Hypo
        INFERENCE CATEGORY :    80
NAME :  Target_Assignment
        Comments : If a third target is being assigned, copy the target value
         and update the aircraft total.

CONDITIONS :
    IsNot       rec1curMap.tgt4 KNOWN
    Name        REC1WMEM.curTarget rec1curMap.tgt4
    Name        rec1curMap.actot+3 rec1curMap.actot
HYPOTHESIS :    rec1AssignTarget.Hypo
        INFERENCE CATEGORY :    70
NAME :  Target_Assignment
        Comments : If a fourth target is being assigned, copy the target value
          and update the aircraft total.

CONDITIONS :
    IsNot       rec1curMap.tgt1 KNOWN
    Name        REC1WMEM.curTarget rec1curMap.tgt1
    Name        rec1curMap.actot+3 rec1curMap.actot
HYPOTHESIS :    rec1AssignTarget.Hypo
        INFERENCE CATEGORY :    100
NAME :  Target_Assignment
        Comments : If the first target is being assigned, copy the target
         value and update the aircraft total.
```

```
CONDITIONS :
    IsNot       rec1curMap.tgt2 KNOWN
    Name        REC1WMEM.curTarget rec1curMap.tgt2
    Name        rec1curMap.actot+3 rec1curMap.actot
HYPOTHESIS :    rec1AssignTarget.Hypo
        INFERENCE CATEGORY :    90
NAME :  Target_Assignment
        Comments : If a second target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    IsNot       rec1curMap.tgt5 KNOWN
    Name        REC1WMEM.curTarget rec1curMap.tgt5
    Name        rec1curMap.actot+3 rec1curMap.actot
    Name        TRUE rec1curMap.doAssign
HYPOTHESIS :    rec1AssignTarget.Hypo
        INFERENCE CATEGORY :    60
NAME :  Target_Assignment

CONDITIONS :
    Is          REC1WMEM.targCursor KNOWN
    =           REC1WMEM.targCursor -1
    No          <|rec1Supply|>.Processed
    Name        TRUE <|rec1Supply|>.dorec1Mapping
HYPOTHESIS :    rec1FinalCheck.Hypo
        INFERENCE CATEGORY :    1
NAME :  Stop_Assignment_rec1Mappings
        Comments : If REC1WMEM.targCursor is equal to -1 then the end of the
         'reckn' database has been reached and there are no more targets to be
          assigned.

CONDITIONS :
    Reset       rec1FinalCheck.Hypo
    Name        rec1FinalCheck.Hypo rec1FinalCheck.Hypo
    >           rec1curMap.actot 0
    Name        TRUE rec1curMap.doAssign
HYPOTHESIS :    rec1Mapping.Hypo
        INFERENCE CATEGORY :    50
NAME :  Stop_Assignment_Loop
        Comments : Since rec1Mapping.Hypo will not be reset, this rule will
         always terminate the loop.  It is only evaluated when all possible
         assignments from the 'AC_XX' object have been done.
```

```
CONDITIONS :
    >=          REC1WMEM.actot 3
    Retrieve    "reckn.nxp" @TYPE=NXPDB;
      @SLOTS=REC1WMEM.curTarget;@FIELDS="abid";@CURSOR=REC1WMEM.targCursor;
    <>          REC1WMEM.targCursor -1
    Name        rec1AssignTarget.Hypo rec1AssignTarget.Hypo
    Name        REC1WMEM.actot-3 REC1WMEM.actot
HYPOTHESIS :    rec1Mapping.Hypo
ACTIONS :
    Reset       rec1Mapping.Hypo
        INFERENCE CATEGORY :    100
NAME :  Define_Assignment_Loop
        Comments : This rule will loop until there are less than three
         available planes from the current 'AC_XX' object.  If there are more
         than 3, then get the next target and assign 3 planes to it.

CONDITIONS :
    IsNot       rec2curMap.tgt5 KNOWN
    Name        REC2WMEM.curTarget rec2curMap.tgt5
    Name        rec2curMap.actot+3 rec2curMap.actot
    Name        TRUE rec2curMap.doAssign
HYPOTHESIS :    rec2AssignTarget.Hypo
        INFERENCE CATEGORY :    60
NAME :  Target_Assignment
        Comments : If the fifth target is being assigned, copy the target
         value, update the aircraft count and trigger creation of a
         new assignment object.

CONDITIONS :
    IsNot       rec2curMap.tgt2 KNOWN
    Name        REC2WMEM.curTarget rec2curMap.tgt2
    Name        rec2curMap.actot+3 rec2curMap.actot
HYPOTHESIS :    rec2AssignTarget.Hypo
        INFERENCE CATEGORY :    90
NAME :  Target_Assignment
        Comments : If a second target is being assigned, copy the target value
         and update the aircraft total.
```

```
CONDITIONS :
    IsNot       rec2curMap.tgt1 KNOWN
    Name        REC2WMEM.curTarget rec2curMap.tgt1
    Name        rec2curMap.actot+3 rec2curMap.actot
HYPOTHESIS :    rec2AssignTarget.Hypo
        INFERENCE CATEGORY :    100
NAME :  Target_Assignment
        Comments : If the first target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    IsNot       rec2curMap.tgt4 KNOWN
    Name        REC2WMEM.curTarget rec2curMap.tgt4
    Name        rec2curMap.actot+3 rec2curMap.actot
HYPOTHESIS :    rec2AssignTarget.Hypo
        INFERENCE CATEGORY :    70
NAME :  Target_Assignment
        Comments : If a fourth target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    IsNot       rec2curMap.tgt3 KNOWN
    Name        REC2WMEM.curTarget rec2curMap.tgt3
    Name        rec2curMap.actot+3 rec2curMap.actot
HYPOTHESIS :    rec2AssignTarget.Hypo
        INFERENCE CATEGORY :    80
NAME :  Target_Assignment
        Comments : If a third target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    Is          REC2WMEM.targCursor KNOWN
    =           REC2WMEM.targCursor -1
    No          <|rec2Supply|>.Processed
    Name        TRUE <|rec2Supply|>.dorec2Mapping
HYPOTHESIS :    rec2FinalCheck.Hypo
        INFERENCE CATEGORY :    1
NAME :  Stop_Assignment_rec2Mappings
        Comments : If REC2WMEM.targCursor is equal to -1 then the end of the
         'reckn' database has been reached and there are no more targets to
         be assigned.  Thus we should stop evaluation of any OS metaslots
         attached to objects under 'rec2Supply' that have not yet been
         triggered.
```

```
CONDITIONS :
    Retrieve    "rec2ac.nxp" @TYPE=NXPDB;@FILL=ADD;
      @NAME="'AC_'!Indx!";@CREATE=|rec2Supply|;@PROPS=actyp,actot;
      @FIELDS="actyp","qnty";
    Name        <|rec2Supply|>.dorec2Mapping <|rec2Supply|>.dorec2Mapping
    >           LENGTH(<|rec2Assignments|>) 0
HYPOTHESIS :    rec2GetPackage.Hypo
ACTIONS :
    Write       "rec2ms.nxp" @TYPE=NXPDB;@FILL=NEW;
      @NAME="'Assign_'!Indx(8)!";@PROPS=actyp,actot,tgt1,tgt2,tgt3,tgt4,tgt5;
      @FIELDS="actyp(10)","qnty(8)","tgt1(7)","tgt2(7)","tgt3(7)",
            "tgt4(7)","tgt5(7)";@ATOMS=<|rec2Assignments|>;
    Write       "rec3ac.nxp" @TYPE=NXPDB;@FILL=NEW;@NAME="'AC_'!Indx(8)!";
      @PROPS=actyp,actot;@FIELDS="actyp(10)","qnty(8)";@ATOMS=<|rec2Supply|>;
        INFERENCE CATEGORY :    1
NAME :  Top_Level_Package_Control
        Comments : This rule represents the top level control structure: Build
         the table of planes to be assigned.  Assign planes (map them on) to
         available targets.  If any assignments have been made, update the
         database tables.

CONDITIONS :
    Reset       rec2FinalCheck.Hypo
    Name        rec2FinalCheck.Hypo rec2FinalCheck.Hypo
    >           rec2curMap.actot 0
    Name        TRUE rec2curMap.doAssign
HYPOTHESIS :    rec2Mapping.Hypo
        INFERENCE CATEGORY :    50
NAME :  Stop_Assignment_Loop
        Comments : Since rec2Mapping.Hypo will not be reset, this rule will
         always terminate the loop.  It is only evaluated when all possible
         assignments from the 'AC_XX' object have been done.  If 'rec2curMap'
         is partially filled it triggers creation of a new assignment objec
```

```
CONDITIONS :
    >=          REC2WMEM.actot 6
    Retrieve    "recunkn.nxp" @TYPE=NXPDB;@SLOTS=REC2WMEM.curTarget;
      @FIELDS="abid";@CURSOR=REC2WMEM.targCursor;
    <>          REC2WMEM.targCursor -1
    Name        rec2AssignTarget.Hypo rec2AssignTarget.Hypo
    Name        REC2WMEM.actot-6 REC2WMEM.actot
HYPOTHESIS :    rec2Mapping.Hypo
ACTIONS :
    Reset       rec2Mapping.Hypo
        INFERENCE CATEGORY :    100
NAME :  Define_Assignment_Loop
        Comments : This rule will loop until there are less than three
         available planes from the current 'AC_XX' object.  If there are more
         than 6, then get the next target and assign 6 planes to it.

CONDITIONS :
    IsNot       currec3Map.tgt5 KNOWN
    Name        REC3WMEM.curTarget currec3Map.tgt5
    Name        currec3Map.actot+3 currec3Map.actot
    Name        TRUE currec3Map.doAssign
HYPOTHESIS :    rec3AssignTarget.Hypo
        INFERENCE CATEGORY :    60
NAME :  Target_Assignment
        Comments : If the fifth target is being assigned, copy the target
         value, update the aircraft count and trigger creation of a
         new assignment object.

CONDITIONS :
    IsNot       currec3Map.tgt2 KNOWN
    Name        REC3WMEM.curTarget currec3Map.tgt2
    Name        currec3Map.actot+3 currec3Map.actot
HYPOTHESIS :    rec3AssignTarget.Hypo
        INFERENCE CATEGORY :    90
NAME :  Target_Assignment
        Comments : If a second target is being assigned, copy the target value
          and update the aircraft total.
```

```
CONDITIONS :
    IsNot       currec3Map.tgt1 KNOWN
    Name        REC3WMEM.curTarget currec3Map.tgt1
    Name        currec3Map.actot+3 currec3Map.actot
HYPOTHESIS :    rec3AssignTarget.Hypo
        INFERENCE CATEGORY :    100
NAME :  Target_Assignment
        Comments : If the first target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    IsNot       currec3Map.tgt4 KNOWN
    Name        REC3WMEM.curTarget currec3Map.tgt4
    Name        currec3Map.actot+3 currec3Map.actot
HYPOTHESIS :    rec3AssignTarget.Hypo
        INFERENCE CATEGORY :    70
NAME :  Target_Assignment
        Comments : If a fourth target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    IsNot       currec3Map.tgt3 KNOWN
    Name        REC3WMEM.curTarget currec3Map.tgt3
    Name        currec3Map.actot+3 currec3Map.actot
HYPOTHESIS :    rec3AssignTarget.Hypo
        INFERENCE CATEGORY :    80
NAME :  Target_Assignment
        Comments : If a third target is being assigned, copy the target
         value and update the aircraft total.

CONDITIONS :
    Is          REC3WMEM.targCursor KNOWN
    =           REC3WMEM.targCursor -1
    No          <|rec3Supply|>.Processed
    Name        TRUE <|rec3Supply|>.dorec3Mapping
HYPOTHESIS :    rec3FinalCheck.Hypo
        INFERENCE CATEGORY :    1
NAME :  Stop_Assignment_rec3Mappings
        Comments : If REC3WMEM.targCursor is equal to -1 then the end of
         the 'recsusp' database has been reached and there are no more targets
         to be assigned.  Thus we should stop evaluation of any OS metaslots
         attached to objects under 'rec3Supply' that have not yet
         been triggered.
```

```
CONDITIONS :
    Reset       rec3FinalCheck.Hypo
    Name        rec3FinalCheck.Hypo rec3FinalCheck.Hypo
    >           currec3Map.actot 0
    Name        TRUE currec3Map.doAssign
HYPOTHESIS :    rec3Mapping.Hypo
        INFERENCE CATEGORY :    50
NAME :  Stop_Assignment_Loop
        Comments : Since rec3Mapping.Hypo will not be reset, this rule will
         always terminate the loop.  It is only evaluated when all possible
         assignments from the 'AC_XX' object have been done.  If 'currec3Map'
         is partially filled it triggers creation of a new assignment objec

CONDITIONS :
    >=          REC3WMEM.actot 9
    Retrieve    "recsusp.nxp" @TYPE=NXPDB;@SLOTS=REC3WMEM.curTarget;
      @FIELDS="abid";@CURSOR=REC3WMEM.targCursor;
    <>          REC3WMEM.targCursor -1
    Name        rec3AssignTarget.Hypo rec3AssignTarget.Hypo
    Name        REC3WMEM.actot-9 REC3WMEM.actot
HYPOTHESIS :    rec3Mapping.Hypo
ACTIONS :
    Reset       rec3Mapping.Hypo
        INFERENCE CATEGORY :    100
NAME :  Define_Assignment_Loop
        Comments : This rule will loop until there are less than nine
          available planes from the current 'AC_XX' object.  If there
          are more than 9, then get the next target and assign 9 planes to it.
```

Appendix D. *Objects*

D.1 Introduction

The objects used in the rule-base are presented in this appendix. Object names indicate the task they are associated with. Objects containing "lr" or "lroca"are used in the long range offensive counter air task. Objects containing "dsup" are used in the defense suppression task. Those containing "rec1," "rec2," "rec3," "recce1," "recce2," or "recce3," pertain to the recconaisance task.

D.2 The Objects

```
NAME :  ataf
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            Hypothesis of Rule 72

NAME :  curlrMap
PROPERTIES :
    actc  = (I) Unknown
NAME :  curlrMap.actot
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
```

```
IF CHANGE DO:
Used In :
    LHS or RHS in Rule 80 (Occurrences: 1)
    LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
    LHS or RHS in Rule Target_Assignment(#98) (Occurrences: 2)
    LHS or RHS in Rule Target_Assignment(#97) (Occurrences: 2)
    LHS or RHS in Rule Target_Assignment(#96) (Occurrences: 2)
    LHS or RHS in Rule Target_Assignment(#95) (Occurrences: 2)
    LHS or RHS in Rule Target_Assignment(#94) (Occurrences: 2)
    LHS or RHS in Rule Stop_Assignment_Loop(#111) (Occurrences: 1)
    Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
Used In :
    LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
    LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
    LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
    LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
    Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
    Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
    Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
    Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    Member Order of Sources of rec3Supply.dorec3Mapping
     (Occurrences: 2)
    Member Order of Sources of rec2Supply.dorec2Mapping
     (Occurrences: 2)
    Member Order of Sources of rec1Supply.dorec1Mapping
     (Occurrences: 2)
    Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
    actyp = (S) Unknown
        Used In :
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Memoer If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
    doAssign = (B) Unknown
NAME :  curlrMap.doAssign
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        Comments : Create a new assignment object, copy values from 'curlrMap'
         into it, and reset the slots of 'curlrMap', it is not necessary to
         reset 'curlrMap.actype'
        ORDER OF SOURCES :
```

```
IF CHANGE DO :
    Reset       curlrMap.doAssign
    Do          LRWMEM.targetIndex+1 LRWMEM.targetIndex
    CreateObjec 'Assign_'\LRWMEM.targetIndex\ IlrAssignments|
    Do          curlrMap.actyp 'Assign_'\LRWMEM.targetIndex\.actyp
    Do          curlrMap.actot 'Assign_'\LRWMEM.targetIndex\.actot
    Do          curlrMap.tgt1 'Assign_'\LRWMEM.targetIndex\.tgt1
    Do          curlrMap.sort1 'Assign_'\LRWMEM.targetIndex\.sort1
    Do          curlrMap.tgt2 'Assign_'\LRWMEM.targetIndex\.tgt2
    Do          curlrMap.sort2 'Assign_'\LRWMEM.targetIndex\.sort2
    Do          curlrMap.tgt3 'Assign_'\LRWMEM.targetIndex\.tgt3
    Do          curlrMap.sort3 'Assign_'\LRWMEM.targetIndex\.sort3
    Do          curlrMap.tgt4 'Assign_'\LRWMEM.targetIndex\.tgt4
    Do          curlrMap.sort4 'Assign_'\LRWMEM.targetIndex\.sort4
    Do          curlrMap.tgt5 'Assign_'\LRWMEM.targetIndex\.tgt5
    Do          curlrMap.sort5 'Assign_'\LRWMEM.targetIndex\.sort5
    Do          curlrMap.ecmtyp 'Assign_'\LRWMEM.targetIndex\.ecmtyp
    Do          curlrMap.ecmtot 'Assign_'\LRWMEM.targetIndex\.ecmtot
    Reset       curlrMap.ecmtot
    Reset       curlrMap.actot
    Reset       curlrMap.tgt1
    Reset       curlrMap.tgt2
    Reset       curlrMap.tgt3
    Reset       curlrMap.tgt4
    Reset       curlrMap.tgt5
    Reset       curlrMap.sort1
    Reset       curlrMap.sort2
    Reset       curlrMap.sort3
    Reset       curlrMap.sort4
    Reset       curlrMap.sort5
Used In :
    LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
    LHS or RHS in Rule Target_Assignment(#98) (Occurrences: 1)
    LHS or RHS in Rule Stop_Assignment_Loop(#111) (Occurrences: 1)
    Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
```

```
    ecmtot =    (I) Unknown
NAME :  curlrMap.ecmtot
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#98) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#97) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#96) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#95) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#94) (Occurrences: 2)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 1)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule 109 (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
    ecmtyp = (S) Unknown
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 1)
            LHS or RHS in Rule Start_lrECM_table_use(#77) (Occurrences: 1)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 1)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule 109 (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
```

```
    sort1 = (I) Unknown
NAME :  curlrMap.sort1
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#97) (Occurrences: 1)
            LHS or RHS in Rule Target_1_assignment(#107) (Occurrences: 1)
            LHS or RHS in Rule Null_target_1(#108) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
    sort2 = (I) Unknown
NAME :  curlrMap.sort2
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#94) (Occurrences: 1)
            LHS or RHS in Rule Target_2_assignment(#106) (Occurrences: 1)
            LHS or RHS in Rule Target_2_assignment(#105) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
```

```
    sort3 = (I) Unknown
NAME :  curlrMap.sort3
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#96) (Occurrences: 1)
            LHS or RHS in Rule Target_3_assignment(#104) (Occurrences: 1)
            LHS or RHS in Rule Target_3_assignment(#103) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
    sort4 = (I) Unknown
NAME :  curlrMap.sort4
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#95) (Occurrences: 1)
            LHS or RHS in Rule Target_4_assignment(#102) (Occurrences: 1)
            LHS or RHS in Rule Target_4_assignment(#101) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
```

```
    sort5 = (I) Unknown
NAME :  curlrMap.sort5
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#98) (Occurrences: 1)
            LHS or RHS in Rule Target_5_assignment(#100) (Occurrences: 1)
            LHS or RHS in Rule Target_5_assignment(#99) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
    status = (F) Unknown
    tgt1 = (I) Unknown
NAME :  curlrMap.tgt1
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
```

```
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#97) (Occurrences: 2)
            LHS or RHS in Rule Target_1_assignment(#107) (Occurrences: 2)
            LHS or RHS in Rule Null_target_1(#108) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt2 = (I) Unknown
NAME :  curlrMap.tgt2
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#94) (Occurrences: 2)
            LHS or RHS in Rule Target_2_assignment(#106) (Occurrences: 2)
            LHS or RHS in Rule Target_2_assignment(#105) (Occurrences: 2)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
    tgt3 = (I) Unknown
NAME :  curlrMap.tgt3
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#96) (Occurrences: 2)
            LHS or RHS in Rule Target_3_assignment(#104) (Occurrences: 2)
            LHS or RHS in Rule Target_3_assignment(#103) (Occurrences: 2)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt4 = (I) Unknown
NAME :  curlrMap.tgt4
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
```

```
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#95) (Occurrences: 2)
            LHS or RHS in Rule Target_4_assignment(#102) (Occurrences: 2)
            LHS or RHS in Rule Target_4_assignment(#101) (Occurrences: 2)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt5 = (I) Unknown
NAME :  curlrMap.tgt5
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#98) (Occurrences: 2)
            LHS or RHS in Rule Target_5_assignment(#100) (Occurrences: 2)
            LHS or RHS in Rule Target_5_assignment(#99) (Occurrences: 2)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
NAME :  currec3Map
PROPERTIES :
    actot = (I) Unknown
NAME :  currec3Map.actot
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#139) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#138) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#137) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#136) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#135) (Occurrences: 2)
            LHS or RHS in Rule Stop_Assignment_Loop(#141) (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 4)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
    actyp = (S) Unknown
        Used In :
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
    doAssign = (B) Unknown
NAME :  currec3Map.doAssign
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        IN_RITANCE STRATEGY :  Breadth first

        Comments : Create a new assignment object, copy values from
         'currec3Map' into it, and reset the slots of 'currec3Map', it is
         not necessary to reset 'currec3Map.actype'
        ORDER OF SOURCES :
```

```
IF CHANGE DO :
    Reset       currec3Map.doAssign
    Do          REC3WMEM.targetIndex+1 REC3WMEM.targetIndex
    CreateObjec 'Assign_'\REC3WMEM.targetIndex\ |rec3Assignments|
    Do          currec3Map.actyp 'Assign_'\REC3WMEM.targetIndex\.actyp
    Do          currec3Map.actot 'Assign_'\REC3WMEM.targetIndex\.actot
    Do          currec3Map.tgt1 'Assign_'\REC3WMEM.targetIndex\.tgt1
    Do          currec3Map.tgt2 'Assign_'\REC3WMEM.targetIndex\.tgt2
    Do          currec3Map.tgt3 'Assign_'\REC3WMEM.targetIndex\.tgt3
    Do          currec3Map.tgt4 'Assign_'\REC3WMEM.targetIndex\.tgt4
    Do          currec3Map.tgt5 'Assign_'\REC3WMEM.targetIndex\.tgt5
    Do          REC3WMEM.targetIndex+1 REC3WMEM.targetIndex
    CreateObjec 'Assign_'\REC3WMEM.targetIndex\ |rec3Assignments|
    Do          currec3Map.actyp 'Assign_'\REC3WMEM.targetIndex\.actyp
    Do          currec3Map.actot 'Assign_'\REC3WMEM.targetIndex\.actot
    Do          currec3Map.tgt1 'Assign_'\REC3WMEM.targetIndex\.tgt1
    Do          currec3Map.tgt2 'Assign_'\REC3WMEM.targetIndex\.tgt2
    Do          currec3Map.tgt3 'Assign_'\REC3WMEM.targetIndex\.tgt3
    Do          currec3Map.tgt4 'Assign_'\REC3WMEM.targetIndex\.tgt4
    Do          currec3Map.tgt5 'Assign_'\REC3WMEM.targetIndex\.tgt5
    Do          REC3WMEM.targetIndex+1 REC3WMEM.targetIndex
    CreateObjec 'Assign_'\REC3WMEM.targetIndex\ |rec3Assignments|
    Do          currec3Map.actyp 'Assign_'\REC3WMEM.targetIndex\.actyp
    Do          currec3Map.actot 'Assign_'\REC3WMEM.targetIndex\.actot
    Do          currec3Map.tgt1 'Assign_'\REC3WMEM.targetIndex\.tgt1
    Do          currec3Map.tgt2 'Assign_'\REC3WMEM.targetIndex\.tgt2
    Do          currec3Map.tgt3 'Assign_'\REC3WMEM.targetIndex\.tgt3
    Do          currec3Map.tgt4 'Assign_'\REC3WMEM.targetIndex\.tgt4
    Do          currec3Map.tgt5 'Assign_'\REC3WMEM.targetIndex\.tgt5
    Reset       currec3Map.actot
    Reset       currec3Map.tgt1
    Reset       currec3Map.tgt2
    Reset       currec3Map.tgt3
    Reset       currec3Map.tgt4
    Reset       currec3Map.tgt5
Used In :
    LHS or RHS in Rule Target_Assignment(#139) (Occurrences: 1)
    LHS or RHS in Rule Stop_Assignment_Loop(#141) (Occurrences: 1)
    Member If Change Actions of currec3Map.doAssign (Occurrences: 1)
```

```
    tgt1 = (I) Unknown
NAME :  currec3Map.tgt1
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#137) (Occurrences: 2)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 4)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt2 = (I) Unknown
NAME :  currec3Map.tgt2
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
```

```
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#138) (Occurrences: 2)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 4)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt3 = (I) Unknown
NAME :  currec3Map.tgt3
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#135) (Occurrences: 2)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 4)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
    tgt4 = (I) Unknown
NAME :  currec3Map.tgt4
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#136) (Occurrences: 2)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 4)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt5 = (I) Unknown
NAME :  currec3Map.tgt5
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
```

```
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#139) (Occurrences: 2)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 4)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)

NAME :  dspglobal
PROPERTIES :
    qnty = (I) 0
"       dspglobal.qnty
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            InitValue   0          IF CHANGE DO :
        Used In :
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 4)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 4)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 1)
            Member Order of Sources of dsupac.addprop (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 1)
            Member Order of Sources of dsupac.addprop (Occurrences: 2)
```

```
    rmdr = (I) 0
NAME :  dspglobal.rmdr
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            InitValue   0          IF CHANGE DO :
        Used In :
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
    sortqnty = (I) 0
NAME :  dspglobal.sortqnty
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            InitValue   0          IF CHANGE DO :
        Used In :
            LHS or RHS in Rule select_dsupac_2(#74)     "rrences: 5)
            LHS or RHS in Rule select_dsupac_1(#75) ..    -ences: 4)
    type = (S) Unknown
        Used In :
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 2)
            Member Order of Sources of dsupac.addprop (Occurrences: 1)
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 2)
            Member Order of Sources of dsupac.addprop (Occurrences: 1)
```

```
NAME :  dsup_missions
PROPERTIES :
    Value = (B) Unknown
        Used In :
            LHS or RHS in Rule 72 (Occurrences: 2)
            Hypothesis of Rule Top_level_dsup(#73)

NAME :  dsupac_commit
PROPERTIES :
    Hypo = (B) Unknown
    Value = (B) Unknown
        Used In :
            Member Order of Sources of dsupac.addprop (Occurrences: 3)
            Hypothesis of Rule select_dsupac_2(#74)
            Hypothesis of Rule select_dsupac_1(#75)
            Hypothesis of Rule select_dsupac_0(#76)

NAME :  dsupqnty
PROPERTIES :
    Value = (I) 100
NAME :  dsupqnty
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            InitValue   100         IF CHANGE DO :
        Used In :
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 4)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 5)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 1)
```

```
NAME :  GET_lrECM
PROPERTIES :
    Hypo = (B) Unknown

NAME :  Get_lrECM_ac
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 3)
            Hypothesis of Rule 80
            Hypothesis of Rule Retrieve_ecm_ac_from_table(#78)
            Hypothesis of Rule Get_ecm_ac_from_same_entry(#79)
            Hypothesis of Rule Start_lrECM_table_use(#77)

NAME :  Get_lrECM_msnsz
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 3)
            Hypothesis of Rule Sufficient_lrECM_msnsz(#82)
            Hypothesis of Rule 81

NAME :  Get_lrECM_needed
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Sufficient_lrECM_msnsz(#82) (Occurrences: 2)
            LHS or RHS in Rule 81 (Occurrences: 2)
            Hypothesis of Rule Non_mission_lrECM_msnsz(#84)
            Hypothesis of Rule Standard_lrECM_msnsz(#83)
```

```
NAME :  Get_lrECM_type
PROPERTIES :
    Hypo = (B) Unknown

NAME :  Get_lrtarget
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 2)
            Hypothesis of Rule 86
            Hypothesis of Rule Find_the_target(#85)
    Value = (B) Unknown

NAME :  GetlrPackage
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule 72 (Occurrences: 2)
            Hypothesis of Rule Top_Level_Package_Control(#87)

NAME :  Getrec1Package
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule 72 (Occurrences: 2)
            Hypothesis of Rule Top_Level_Package_Control(#88)

NAME :  Getrec3Package
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule 72 (Occurrences: 2)
            Hypothesis of Rule Top_Level_Package_Control(#89)
```

```
NAME :  lrACneeded
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 1)
            LHS or RHS in Rule Determine_the_mission_size_when_there_
             are_essentially_no_ac_left(#113) (Occurrences: 2)
            LHS or RHS in Rule Determine_mission_size_when_there_
             are_minimal_ac(#114) (Occurrences: 2)
            LHS or RHS in Rule Determine_mission_size_when_there_
               are_ample_ac(#115) (Occurrences: 2)
            LHS or RHS in Rule Determine_mission_size_for_target(#116)
             (Occurrences: 2)
            LHS or RHS in Rule Determine_mission_size_for_last_target(#117)
              (Occurrences: 2)
            Hypothesis of Rule Find_AC_needed_75(#90)
            Hypothesis of Rule Find_AC_needed_25(#92)
            Hypothesis of Rule Find_AC_needed_50(#91)
            Hypothesis of Rule Find_AC_needed_100(#93)

NAME :  lrAssignTarget
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 2)
            Hypothesis of Rule Target_Assignment(#98)
            Hypothesis of Rule Target_Assignment(#97)
            Hypothesis of Rule Target_Assignment(#96)
            Hypothesis of Rule Target_Assignment(#95)
            Hypothesis of Rule Target_Assignment(#94)
            Hypothesis of Rule Target_4_assignment(#101)
            Hypothesis of Rule Target_3_assignment(#103)
            Hypothesis of Rule Null_target_1(#108)
            Hypothesis of Rule Target_1_assignment(#107)
            Hypothesis of Rule Target_2_assignment(#106)
            Hypothesis of Rule Target_3_assignment(#104)
            Hypothesis of Rule Target_4_assignment(#102)
            Hypothesis of Rule Target_5_assignment(#99)
```

```
NAME :  lrECM_table
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
            LHS or RHS in Rule 109 (Occurrences: 3)
            Hypothesis of Rule 109

NAME :  lrFinalCheck
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Stop_Assignment_Loop(#111) (Occurrences: 3)
            Hypothesis of Rule Stop_Assignment_lrMappings(#110)

NAME :  lrHold
PROPERTIES :
    curstatus = (F) Unknown
NAME :  lrHold.curstatus
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0.0
        IF CHANGE DO :
```

```
    status = (F) Unknown
NAME :  lrHold.status
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0.0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule 86 (Occurrences: 2)
            LHS or RHS in Rule Target_5_assignment(#100) (Occurrences: 1)
            LHS or RHS in Rule Target_4_assignment(#101) (Occurrences: 1)
            LHS or RHS in Rule Target_3_assignment(#103) (Occurrences: 1)
            LHS or RHS in Rule Target_2_assignment(#105) (Occurrences: 1)
            LHS or RHS in Rule Null_target_1(#108) (Occurrences: 1)
    target = (I) Unknown
NAME :  lrHold.target
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
```

```
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule 86 (Occurrences: 3)
            LHS or RHS in Rule Find_the_target(#85) (Occurrences: 1)
            LHS or RHS in Rule Target_5_assignment(#100) (Occurrences: 2)
            LHS or RHS in Rule Target_5_assignment(#99) (Occurrences: 1)
            LHS or RHS in Rule Target_4_assignment(#102) (Occurrences: 1)
            LHS or RHS in Rule Target_4_assignment(#101) (Occurrences: 2)
            LHS or RHS in Rule Target_3_assignment(#104) (Occurrences: 1)
            LHS or RHS in Rule Target_3_assignment(#103) (Occurrences: 2)
            LHS or RHS in Rule Target_2_assignment(#106) (Occurrences: 1)
            LHS or RHS in Rule Target_2_assignment(#105) (Occurrences: 2)
            LHS or RHS in Rule Target_1_assignment(#107) (Occurrences: 1)
            LHS or RHS in Rule Null_target_1(#108) (Occurrences: 2)

NAME :  lrMapping
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 3)
            Hypothesis of Rule Define_Assignment_Loop(#112)
            Hypothesis of Rule Stop_Assignment_Loop(#111)

NAME :  lrMission_size
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 3)
            Hypothesis of Rule
              Determine_the_mission_size_when_there_are_essentially_
              no_ac_left(#113)
            Hypothesis of Rule
              Determine_mission_size_when_there_are_minimal_ac(#114)
            Hypothesis of Rule Determine_mission_size_for_last_target(#117)
            Hypothesis of Rule Determine_mission_size_for_target(#116)
            Hypothesis of Rule Determine_mission_size_when_
              there_are_ample_ac(#115)
```

```
NAME :  LRWMEM
PROPERTIES :
    acneeded = (I) Unknown
        Used In :
            LHS or RHS in Rule Find_AC_needed_75(#90) (Occurrences: 1)
            LHS or RHS in Rule Find_AC_needed_50(#91) (Occurrences: 1)
            LHS or RHS in Rule Find_AC_needed_25(#92) (Occurrences: 1)
            LHS or RHS in Rule Find_AC_needed_100(#93) (Occurrences: 1)
            LHS or RHS in Rule
             Determine_the_mission_size_when_there_are_essentially_no_
              ac_left(#113) (Occurrences: 1)
            LHS or RHS in Rule
              Determine_mission_size_when_there_are_
               minimal_ac(#114) (Occurrences: 1)
            LHS or RHS in Rule
               Determine_mission_size_when_there_are_
                ample_ac(#115) (Occurrences: 2)
            LHS or RHS in Rule Determine_mission_size_for_target(#116)
               (Occurrences: 1)
            LHS or RHS in Rule Determine_mission_size_for_last_target(#117)
               (Occurrences: 1)
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 3)
            LHS or RHS in Rule
             Determine_the_mission_size_when_there_are_
              essentially_no_ac_left(#113) (Occurrences: 2)
            LHS or RHS in Rule
             Determine_mission_size_when_there_are_
              minimal_ac(#114) (Occurrences: 3)
            LHS or RHS in Rule
             Determine_mission_size_when_there_are_ample_ac(#115)
              (Occurrences: 1)
            LHS or RHS in Rule Determine_mission_size_for_target(#116)
              (Occurrences: 2)
            LHS or RHS in Rule Determine_mission_size_for_last_target(#117)
              (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
        Member Order of Sources of rec3Supply.dorec3Mapping
         (Occurrences: 2)
        Member Order of Sources of rec2Supply.dorec2Mapping
         (Occurrences: 2)
        Member Order of Sources of rec1Supply.dorec1Mapping
         (Occurrences: 2)
        Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)

curstatus = (F) Unknown
    Used In :
        LHS or RHS in Rule 86 (Occurrences: 1)
        LHS or RHS in Rule Find_the_target(#85) (Occurrences: 1)
        LHS or RHS in Rule Find_AC_needed_75(#90) (Occurrences: 2)
        LHS or RHS in Rule Find_AC_needed_50(#91) (Occurrences: 2)
        LHS or RHS in Rule Find_AC_needed_25(#92) (Occurrences: 2)
        LHS or RHS in Rule Find_AC_needed_100(#93) (Occurrences: 1)
        LHS or RHS in Rule Target_5_assignment(#100) (Occurrences: 1)
        LHS or RHS in Rule Target_3_assignment(#103) (Occurrences: 1)
        LHS or RHS in Rule Null_target_1(#108) (Occurrences: 1)
curTarget = (I) Unknown
    Used In :
        LHS or RHS in Rule 86 (Occurrences: 1)
        LHS or RHS in Rule Find_the_target(#85) (Occurrences: 1)
        LHS or RHS in Rule Target_Assignment(#97) (Occurrences: 1)
        LHS or RHS in Rule Target_Assignment(#96) (Occurrences: 1)
        LHS or RHS in Rule Target_Assignment(#95) (Occurrences: 1)
        LHS or RHS in Rule Target_Assignment(#94) (Occurrences: 1)
        LHS or RHS in Rule Target_5_assignment(#100) (Occurrences: 1)
        LHS or RHS in Rule Target_4_assignment(#101) (Occurrences: 1)
        LHS or RHS in Rule Target_2_assignment(#105) (Occurrences: 1)
        LHS or RHS in Rule Null_target_1(#108) (Occurrences: 2)
```

```
    didg = (F) Unknown
        Used In :
            LHS or RHS in Rule Find_AC_needed_75(#90) (Occurrences: 1)
            LHS or RHS in Rule Find_AC_needed_50(#91) (Occurrences: 1)
            LHS or RHS in Rule Find_AC_needed_25(#92) (Occurrences: 1)
            LHS or RHS in Rule Find_AC_needed_100(#93) (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
    ecmCursor = (I) Unknown
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 1)
            LHS or RHS in Rule Start_lrECM_table_use(#77) (Occurrences: 1)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
            LHS or RHS in Rule 109 (Occurrences: 3)
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 1)
    ecmindx = (I) 0
NAME :  LRWME%.ecmindx
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            InitValue   0         IF CHANGE DO :
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 5)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 5)
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 5)
            LHS or RHS in Rule 109 (Occurrences: 5)
```

```
ecmmsnsz = (I) Unknown
    Used In :
        LHS or RHS in Rule Sufficient_lrECM_msnsz(#82) (Occurrences: 1)
        LHS or RHS in Rule 81 (Occurrences: 1)
        LHS or RHS in Rule Target_Assignment(#98) (Occurrences: 1)
        LHS or RHS in Rule Target_Assignment(#97) (Occurrences: 1)
        LHS or RHS in Rule Target_Assignment(#95) (Occurrences: 1)
        LHS or RHS in Rule Target_Assignment(#94) (Occurrences: 1)
        LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 1)
ecmneeded = (I) Unknown
    Used In :
        LHS or RHS in Rule Sufficient_lrECM_msnsz(#82) (Occurrences: 2)
        LHS or RHS in Rule 81 (Occurrences: 1)
        LHS or RHS in Rule Standard_lrECM_msnsz(#83) (Occurrences: 1)
        LHS or RHS in Rule Non_mission_lrECM_msnsz(#84) (Occurrences: 1)
ecmtot = (I) Unknown
    Used In :
        LHS or RHS in Rule 80 (Occurrences: 3)
        LHS or RHS in Rule Start_lrECM_table_use(#77) (Occurrences: 1)
        LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 3)
        LHS or RHS in Rule Get_ecm_ac_from_same_entry(#79) (Occurrences: 1)
        LHS or RHS in Rule Sufficient_lrECM_msnsz(#82) (Occurrences: 1)
        LHS or RHS in Rule 81 (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule 109 (Occurrences: 2)
        LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 2)
    Used In :
        LHS or RHS in Rule 80 (Occurrences: 1)
        LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
        LHS or RHS in Rule 109 (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
```

```
    ecmtyp = (S) Unknown
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 3)
            LHS or RHS in Rule Start_lrECM_table_use(#77) (Occurrences: 3)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule 109 (Occurrences: 2)
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 1)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule 109 (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)

    efsrt = (F) Unknown
        Used In :
            LHS or RHS in Rule Find_AC_needed_75(#90) (Occurrences: 2)
            LHS or RHS in Rule Find_AC_needed_50(#91) (Occurrences: 2)
            LHS or RHS in Rule Find_AC_needed_25(#92) (Occurrences: 2)
            LHS or RHS in Rule Find_AC_needed_100(#93) (Occurrences: 2)
    msnsz = (I) 0
NAME :  LRWMEM.msnsz
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first
```

```
ORDER OF SOURCES :
    InitValue   0          IF CHANGE DO :
Used In :
    LHS or RHS in Rule 81 (Occurrences: 1)
    LHS or RHS in Rule Standard_lrECM_msnsz(#83) (Occurrences: 1)
    LHS or RHS in Rule Non_mission_lrECM_msnsz(#84) (Occurrences: 1)
    LHS or RHS in Rule Target_Assignment(#98) (Occurrences: 3)
    LHS or RHS in Rule Target_Assignment(#97) (Occurrences: 3)
    LHS or RHS in Rule Target_Assignment(#96) (Occurrences: 3)
    LHS or RHS in Rule Target_Assignment(#95) (Occurrences: 3)
    LHS or RHS in Rule Target_Assignment(#94) (Occurrences: 3)
    LHS or RHS in Rule Target_5_assignment(#100) (Occurrences: 1)
    LHS or RHS in Rule Target_5_assignment(#99) (Occurrences: 1)
    LHS or RHS in Rule Target_4_assignment(#102) (Occurrences: 1)
    LHS or RHS in Rule Target_4_assignment(#101) (Occurrences: 1)
    LHS or RHS in Rule Target_3_assignment(#104) (Occurrences: 1)
    LHS or RHS in Rule Target_3_assignment(#103) (Occurrences: 1)
    LHS or RHS in Rule Target_2_assignment(#106) (Occurrences: 1)
    LHS or RHS in Rule Target_2_assignment(#105) (Occurrences: 1)
    LHS or RHS in Rule Target_1_assignment(#107) (Occurrences: 1)
    LHS or RHS in Rule Null_target_1(#108) (Occurrences: 1)
    LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 1)
    LHS or RHS in Rule
      Determine_the_mission_size_when_there_
       are_essentially_no_ac_left(#113) (Occurrences: 1)
    LHS or RHS in Rule
      Determine_mission_size_when_there_
       are_minimal_ac(#114) (Occurrences: 1)
    LHS or RHS in Rule
      Determine_mission_size_when_
       there_are_ample_ac(#115) (Occurrences: 1)
    LHS or RHS in Rule Determine_mission_size_for_target(#116)
      (Occurrences: 1)
    LHS or RHS in Rule Determine_mission_size_for_last_target(#117)
      (Occurrences: 1)
```

```
    retflag = (B) True
NAME :  LRWMEM.retflag
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            InitValue   TRUE          IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#98) (Occurrences: 1)
            LHS or RHS in Rule Target_Assignment(#97) (Occurrences: 1)
            LHS or RHS in Rule Target_5_assignment(#100) (Occurrences: 1)
            LHS or RHS in Rule Target_5_assignment(#99) (Occurrences: 1)
            LHS or RHS in Rule Target_1_assignment(#107) (Occurrences: 1)
            LHS or RHS in Rule Null_target_1(#108) (Occurrences: 1)
    targCursor = (I) Unknown
        Used In :
            LHS or RHS in Rule Find_the_target(#85) (Occurrences: 1)
            LHS or RHS in Rule Stop_Assignment_lrMappings(#110)
             (Occurrences: 2)
            LHS or RHS in Rule Define_Assignment_Loop(#112) (Occurrences: 1)
            LHS or RHS in Rule
             Determine_the_mission_size_when_there_are_
              essentially_no_ac_left(#113) (Occurrences: 1)
            LHS or RHS in Rule
             Determine_mission_size_when_there_are_minimal_
              ac(#114) (Occurrences: 1)
            LHS or RHS in Rule Determine_mission_size_for_target(#116)
              (Occurrences: 1)
            LHS or RHS in Rule Determine_mission_size_for_last_target(#117)
              (Occurrences: 1)
    target = (I) Unknown
```

```
    targetIndex = (I) Unknown
NAME :  LRWME1.targetIndex
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            Member If Change Actions of curlrMap.doAssign (Occu rences: 17)

NAME :  n
PROPERTIES :
    Value = (I) Unknown
        Used In :
            Member Order of Sources of msac.index (Occurrences: 3)

NAME :  rec1AssignTarget
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#125) (Occurrences: 2)
            Hypothesis of Rule Target_Assignment(#122)
            Hypothesis of Rule Target_Assignment(#121)
            Hypothesis of Rule Target_Assignment(#120)
            Hypothesis of Rule Target_Assignment(#119)
            Hypothesis of Rule Target_Assignment(#118)

NAME :  rec1curMap
PROPERTIES :
    actot = (I) Unknown
```

```
NAME :  rec1curMap.actot
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#122) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#121) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#120) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#119) (Occurrences: 2)
            LHS or RHS in Rule Target_Assignment(#118) (Occurrences: 2)
            LHS or RHS in Rule Stop_Assignment_Loop(#124) (Occurrences: 1)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
actyp = (S) Unknown
    Used In :
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member Order of Sources of rec1Supply.dorec1Mapping
          (Occurrences: 1)
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
        Member Order of Sources of rec3Supply.dorec3Mapping
         (Occurrences: 1)
        Member Order of Sources of rec2Supply.dorec2Mapping
         (Occurrences: 1)
        Member Order of Sources of rec1Supply.dorec1Mapping
         (Occurrences: 1)
        Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
```

```
    doAssign = (B) Unknown
NAME :  rec1curMap.doAssign
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        Comments : Create a new assignment object, copy values from
         'rec1curMap' into it, and reset the slots of 'rec1curMap', it is
         not necessary to reset 'rec1curMap.actype'
        ORDER OF SOURCES :
        IF CHANGE DO :
            Reset       rec1curMap.doAssign
            Do          REC1WMEM.targetIndex+1 REC1WMEM.targetIndex
            CreateObjec 'Assign_'\REC1WMEM.targetIndex\ |rec1Assignments|
            Do         rec1curMap.actyp 'Assign_'\REC1WMEM.targetIndex\.actyp
            Do         rec1curMap.actot 'Assign_'\REC1WMEM.targetIndex\.actot
            Do          rec1curMap.tgt1 'Assign_'\REC1WMEM.targetIndex\.tgt1
            Do          rec1curMap.tgt2 'Assign_'\REC1WMEM.targetIndex\.tgt2
            Do          rec1curMap.tgt3 'Assign_'\REC1WMEM.targetIndex\.tgt3
            Do          rec1curMap.tgt4 'Assign_'\REC1WMEM.targetIndex\.tgt4
            Do          rec1curMap.tgt5 'Assign_'\REC1WMEM.targetIndex\.tgt5
            Reset       rec1curMap.actot
            Reset       rec1curMap.tgt1
            Reset       rec1curMap.tgt2
            Reset       rec1curMap.tgt3
            Reset       rec1curMap.tgt4
            Reset       rec1curMap.tgt5
        Used In :
            LHS or RHS in Rule Target_Assignment(#118) (Occurrences: 1)
            LHS or RHS in Rule Stop_Assignment_Loop(#124) (Occurrences: 1)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
```

```
    tgt1 = (I) Unknown
NAME :  rec1curMap.tgt1
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#120) (Occurrences: 2)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
    tgt2 = (I) Unknown
NAME :  rec1curMap.tgt2
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#119) (Occurrences: 2)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
    tgt3 = (I) Unknown
NAME :  rec1curMap.tgt3
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#122) (Occurrences: 2)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt4 = (I) Unknown
NAME :  rec1curMap.tgt4
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first
```

```
        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#121) (Occurrences: 2)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt5 = (I) Unknown
NAME :  rec1curMap.tgt5
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            RunTimeValu 0
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Target_Assignment(#118) (Occurrences: 2)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
NAME :  rec1FinalCheck
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Stop_Assignment_Loop(#124) (Occurrences: 3)
            Hypothesis of Rule Stop_Assignment_rec1Mappings(#123)

NAME :  rec1Mapping
CLASSES :
    Hypos
PROPERTIES :
    Hypo = (B) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#125) (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 3)
            Hypothesis of Rule Define_Assignment_Loop(#125)
            Hypothesis of Rule Stop_Assignment_Loop(#124)

NAME :  REC1WMEM
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Define_Assignment_Loop(#125) (Occurrences: 3)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
    curTarget = (I) Unknown
        Used In :
            LHS or RHS in Rule Target_Assignment(#122) (Occurrences: 1)
            LHS or RHS in Rule Target_Assignment(#121) (Occurrences: 1)
            LHS or RHS in Rule Target_Assignment(#120) (Occurrences: 1)
            LHS or RHS in Rule Target_Assignment(#119) (Occurrences: 1)
            LHS or RHS in Rule Target_Assignment(#118) (Occurrences: 1)
            LHS or RHS in Rule Define_Assignment_Loop(#125) (Occurrences: 1)
    targCursor = (I) Unknown
        Used In :
            LHS or RHS in Rule Stop_Assignment_rec1Mappings(#123)
             (Occurrence:: 2)
            LHS or RHS in Rule Define_Assignment_Loop(#125) (Occurrences: 2)
    targetIndex = (I) Unknown
NAME :  REC1WMEM.targetIndex
```

Appendix E. *Classes*

E.1 Introduction

The classes of objects used in the automated red player rule-base are shown here. They are given in the format used by Nexpert Object when the classes are written to a file for examination by a programmer.

E.2 The Classes

```
NAME : acr
    Used In :
        LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 1)
        LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 1)
        LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 1)
        LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 1)
PROPERTIES :
    qnty = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 1)
            Member Order of Sources of dsupac.addprop (Occurrences: 2)
    type = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 2)
            Member Order of Sources of dsupac.addprop (Occurrences: 1)
```

```
NAME :  DSPWMEM
PROPERTIES :
    index = (I) 0
NAME :  DSPWMEM.index
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            InitValue   0         IF CHANGE DO :
        Used In :
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 8)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 8)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 6)
        Used In :
            Member Order of Sources of msac.index (Occurrences: 1)
```

```
NAME :  dsupac
    Used In :
        LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 1)
PROPERTIES :
    addprop = (B) Unknown
NAME :  dsupac.addprop
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            Do          SELF.qnty dspglobal.qnty
            Do          SELF.type dspglobal.type
            Reset       dsupac_commit
            Do          dsupac_commit dsupac_commit
            Do          dspglobal.qnty SELF.qnty
            RunTimeValu TRUE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 2) in
             pattern matching
    qnty = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 1)
            Member Order of Sources of dsupac.addprop (Occurrences: 2)
    type = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 2)
            Member Order of Sources of dsupac.addprop (Occurrences: 1)
```

```
NAME :  Hypos
PROPERTIES :
    Hypo = (B) Unknown

NAME :  lrAssignments
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
    actyp = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
```

```
ecmtot = (I) Unknown
    Used In :
        LHS or RHS in Rule 80 (Occurrences: 1)
        LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
        LHS or RHS in Rule 109 (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
ecmtyp = (S) Unknown
    Used In :
        LHS or RHS in Rule 80 (Occurrences: 1)
        LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
        LHS or RHS in Rule 109 (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
sort1 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
sort2 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
sort3 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
sort4 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
sort5 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
```

```
tgt1 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
tgt2 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
tgt3 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
tgt4 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
    tgt5 = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)

NAME :  lrECMac
    Used In :
        LHS or RHS in Rule 80 (Occurrences: 1)
        LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
        LHS or RHS in Rule 109 (Occurrences: 1)
PROPERTIES :
    ecmtot = (I) Unknown
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 1)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule 109 (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
    ecmtyp = (S) Unknown
        Used In :
            LHS or RHS in Rule 80 (Occurrences: 1)
            LHS or RHS in Rule Retrieve_ecm_ac_from_table(#78) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule 109 (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
    Indx = (I) Unknown
```

```
NAME :  lrSupply
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
    actyp = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Membe. If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
```

```
didb = (F) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
didg = (F) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
        Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
didp = (F) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
dinb = (F) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
ding = (F) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
dinp = (F) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 2)
```

```
    dolrMapping = (B) Unknown
NAME :  lrSupply.dolrMapping
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        PROMPT LINE :           Mark the current object as being processed,
          copy aircraft type and totals to temporary objects, trigger
          assignment loop, and update number of available aircraft.
        ORDER OF SOURCES :
            Do          TRUE SELF.Processed
            Do          SELF.actot LRWMEM.actot
            Do          SELF.actyp curlrMap.actyp
            Do          SELF.didg LRWMEM.didg
            Reset       lrMapping.Hypo
            Do          lrMapping.Hypo lrMapping.Hypo
            Do          LRWMEM.actot SELF.actot
            RunTimeValu TRUE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87)
             (Occurrences: 2) in pattern matching
            LHS or RHS in Rule Stop_Assignment_lrMappings(#110)
             (Occurrences: 1) in pattern matching
    Processed = (B) Unknown
NAME :  lrSupply.Processed
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        Comments : If these objects have not yet been visited, then they
          are marked as not being processed.
        ORDER OF SOURCES :
            RunTimeValu FALSE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Stop_Assignment_lrMappings(#110)
             (Occurrences: 1) in pattern matching
```

```
NAME :  msac
    Used In :
        LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 2)
        LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 1)
        LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 1)
PROPERTIES :
    index = (I) Unknown
NAME :  msac.index
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        ORDER OF SOURCES :
            Do          n+1 n
            Do          n SELF.index
        IF CHANGE DO :
        Used In :
            Member Order of Sources of msac.index (Occurrences: 1)
    qnty = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 1)
            Member Order of Sources of dsupac.addprop (Occurrences: 2)
    type = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 2)
            Member Order of Sources of dsupac.addprop (Occurrences: 1)
```

```
NAME :  rec1Assignments
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 2)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
    actyp = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
```

```
tgt1 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
tgt2 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
tgt3 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
    tgt4 = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt5 = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)

NAME :  rec1Supply
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#83) (Occurrences: 2)
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
    actyp = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
    dorec1Mapping = (B) Unknown
NAME :  rec1Supply.dorec1Mapping
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        PROMPT LINE :           Mark the current object as being processed,
          copy aircraft type and totals to temporary objects, trigger
          assignment loop, and update number of available aircraft.
        ORDER OF SOURCES :
            Do          TRUE SELF.Processed
            Do          SELF.actot REC1WMEM.actot
            Do          SELF.actyp rec1curMap.actyp
            Reset       rec1Mapping.Hypo
            Do          rec1Mapping.Hypo rec1Mapping.Hypo
            Do          REC1WMEM.actot SELF.actot
            RunTimeValu TRUE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#88)
             (Occurrences: 2) in pattern matching
            LHS or RHS in Rule Stop_Assignment_rec1Mappings(#123)
             (Occurrences: 1) in pattern matching
```

```
    Processed = (B) Unknown
NAME :  rec1Supply.Processed
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILTTY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        Comments : If these objects have not yet been visited, then they
           are marked as not being processed.
        ORDER OF SOURCES :
            RunTimeValu FALSE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Stop_Assignment_rec1Mappings(#123)
             (Occurrences: 1) in pattern matching

NAME :  rec2Assignments
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 2)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
.ctyp = (S) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
        Member Order of Sources of rec3Supply.dorec3Mapping
         (Occurrences: 1)
        Member Order of Sourc    f rec2Supply.dorec2Mapping
         (Occurrences: 1)
        Member Order of Sources of rec1Supply.dorec1Mapping
         (Occurrences: 1)
        Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
tgt1 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
tgt2 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
tgt3 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
    tgt4 = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt5 = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)

NAME :  rec2Supply
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 2)
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
   actyp = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
            Member Order of Sources of rec2Supply.dorec2Mapping
              (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
    dorec2Mapping = (B) Unknown
NAME :  rec2Supply.dorec2Mapping
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        PROMPT LINE :           Mark the current object as being processed,
          copy aircraft type and totals to temporary objects, trigger
          assignment loop, and update number of available aircraft.
        ORDER OF SOURCES :
            Do          TRUE SELF.Processed
            Do          SELF.actot REC2WMEM.actot
            Do          SELF.actyp rec2curMap.actyp
            Reset       rec2Mapping.Hypo
            Do          rec2Mapping.Hypo rec2Mapping.Hypo
            Do          REC2WMEM.actot SELF.actot
            RunTimeValu TRUE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Stop_Assignment_rec2Mappings(#131)
             (Occurrences: 1) in pattern matching
            LHS or RHS in Rule Top_Level_Package_Control(#132)
             (Occurrences: 2) in pattern matching
```

```
    Processed = (B) Unknown
NAME :  rec2Supply.Processed
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        Comments : If these objects have not yet been visited, then they
         are marked as not being processed.
        ORDER OF SOURCES :
            RunTimeValu FALSE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Stop_Assignment_rec2Mappings(#131)
             (Occurrences: 1) in pattern matching

NAME :  rec3Assignments
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 2)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
actyp = (S) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
        Member Order of Sources of rec3Supply.dorec3Mapping
         (Occurrences: 1)
        Member Order of Sources of rec2Supply.dorec2Mapping
         (Occurrences: 1)
        Member Order of Sources of rec1Supply.dorec1Mapping
         (Occurrences: 1)
        Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
tgt1 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
tgt2 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
tgt3 = (I) Unknown
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
        LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
        Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
        Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
        Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
        Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
```

```
    tgt4 = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
    tgt5 = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 1)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 1)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)

NAME :  rec3Supply
    Used In :
        LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 2)
PROPERTIES :
    actot = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 2)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 2)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 2)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 2)
```

```
    actyp = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#87) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#88) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#89) (Occurrences: 3)
            LHS or RHS in Rule Top_Level_Package_Control(#132) (Occurrences: 3)
            Member If Change Actions of curlrMap.doAssign (Occurrences: 1)
            Member If Change Actions of currec3Map.doAssign (Occurrences: 3)
            Member If Change Actions of rec1curMap.doAssign (Occurrences: 1)
            Member If Change Actions of rec2curMap.doAssign (Occurrences: 2)
            Member Order of Sources of rec3Supply.dorec3Mapping
             (Occurrences: 1)
            Member Order of Sources of rec2Supply.dorec2Mapping
             (Occurrences: 1)
            Member Order of Sources of rec1Supply.dorec1Mapping
             (Occurrences: 1)
            Member Order of Sources of lrSupply.dolrMapping (Occurrences: 1)
    dorec3Mapping = (B) Unknown
NAME :  rec3Supply.dorec3Mapping
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first
        INHERITANCE STRATEGY :  Breadth first
        PROMPT LINE :           Mark the current object as being processed,
          copy aircraft type and totals to temporary objects, trigger
           assignment loop, and update number of available aircraft.
        ORDER OF SOURCES :
            Do          TRUE SELF.Processed
            Do          SELF.actot REC3WMEM.actot
            Do          SELF.actyp currec3Map.actyp
            Reset       rec3Mapping.Hypo
            Do          rec3Mapping.Hypo rec3Mapping.Hypo
            Do          REC3WMEM.actot SELF.actot
            RunTimeValu TRUE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Top_Level_Package_Control(#89)
             (Occurrences: 2) in pattern matching
            LHS or RHS in Rule Stop_Assignment_rec3Mappings(#140)
             (Occurrences: 1) in pattern matching
```

```
    Processed = (B) Unknown
NAME :  rec3Supply.Processed
        INFERENCE CATEGORY :    1
        INHERITANCE CATEGORY :  1
        SLOT INHERITABILITY :   Default
        VALUE INHERITABILITY :  Default
        INHERITANCE STRATEGY :  Default
        INHERITANCE STRATEGY :  Class first

        INHERITANCE STRATEGY :  Breadth first

        Comments : If these objects have not yet been visited, then they
         are marked as not being processed.
        ORDER OF SOURCES :
            RunTimeValu FALSE
        IF CHANGE DO :
        Used In :
            LHS or RHS in Rule Stop_Assignment_rec3Mappings(#140)
             (Occurrences: 1) in pattern matching

NAME :  tempmsac
PROPERTIES :
    index = (I) Unknown
        Used In :
            Member Order of Sources of msac.index (Occurrences: 1)
    qnty = (I) Unknown
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences: 1)
            Member Order of Sources of dsupac.addprop (Occurrences: 2)
    type = (S) Unknown
        Used In :
            LHS or RHS in Rule Top_level_dsup(#73) (Occurrences: 3)
            LHS or RHS in Rule select_dsupac_2(#74) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_1(#75) (Occurrences: 2)
            LHS or RHS in Rule select_dsupac_0(#76) (Occurrences. 2)
            Member Order of Sources of dsupac.addprop (Occurrences: 1)
```

Bibliography

Air87. Air War College. *Theater War Exercise Users' Handbook 1987*, 1987.

Air88. Air War College. *Agile Eagle '88*, 1988.

All87. Thomas B. Allen. *War Games.* McGraw-Hill Book Company, New York, New York, 1987.

Dav84. Paul K. Davis. Rand's experience in applying artificial intelligence techniques to strategic-level military-political war gaming. Technical Report P-6977, The RAND Corporation, Santa Monica, California, 1984.

Dav88. Paul K. Davis. Applying artificial intelligence techniques to strategic-level gaming and simulation. Technical Report N-2752-RC, The RAND Corporation, Santa Monica, California, 1988.

DBK86. Paul K. Davis, Steven C. Bankes, and James P. Kahan. A new methodology for modeling national command level decisionmaking in war games and simulations. Technical Report R-3290-NA, The RAND Corporation, Santa Monica, California, 1986.

DS85. Paul K. Davis and William L. Schwabe. Search for a red agent to be used in war games and simulations. Technical Report P-7107, The RAND Corporation, Santa Monica, California, 1985.

Har89. Capt Harold D. Harken. An expert system for automating nuclear strike aircraft replacement, aircraft beddown, and logistics movement for the theater warfare exercise. Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1989.

LDS89. J. Liebowitz and D. De Salvo. *Structuring Expert Systems: Domain, Design, and Development.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1989.

Ped89. K. Pederson. *Expert Systems Programming: Practical Techniques for Rule-Based Sytems.* John Wiley and Sons Inc., New York, New York, 1989.